N^o d'ordre : 00000 ANNÉE 2014

UNIVERSITÉ DE RENNES 1

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale Matisse

présentée par

Marco BEVILACQUA

préparée au centre de recherche INRIA Rennes - Bretagne Atlantique

---

Algorithms for Super-resolution of Images and Videos based on Learning Methods

Thèse soutenue à Rennes le 4 juin 2014

devant le jury composé de :

Michel KIEFFER
Professeur, Université Paris-Sud 11 / *Président*

Enrico MAGLI
Professeur, Ecole Polytechnique de Turin / *Rapporteur*

Thomas RODET
Professeur, Ecole Normale Supérieure de Cachan / *Rapporteur*

Fabrizio ARGENTI
Professeur, Université de Florence / *Examinateur*

Marie-Line ALBERI MOREL
Chercheuse, Alcatel-Lucent Bell Labs / *Examinatrice*

Aline ROUMY
Chargée de Recherche, INRIA / *Directrice de thèse*

Christine GUILLEMOT
Directeur de Recherche, INRIA / *Co-directrice de thèse*

"Une vie, c'est fait avec l'avenir, comme les corps sont faits avec du vide."
Jean-Paul Sartre

Acknowledgments

I would like, first, to thank Prof. Thomas Rodet and Prof. Enrico Magli for accepting the role of reviewers. An equal big "thanks" goes to Prof. Michel Kieffer and Prof. Fabrizio Argenti, who accepted to be part of this jury as examiners. To all of them, thanks for taking the time to read this manuscript and for all the valuable comments and remarks on it, which helped me clarifying several aspects of the work done.

As members of this jury too, but especially for their precious support in constantly supervising my work at INRIA, I would like to express my deep gratitude to Dr. Aline Roumy, in particular for making this Ph.D. actually possible by first taking into account my candidature and for her extreme helpfulness (special greetings also to her beautiful family); and to Dr. Christine Guillemot, whose experience and very focused advises have been highly helpful to carry out this thesis. As a supervisor at a distance (but only "physical"), I would also like to warmly thank Dr. Marie-Line Alberi Morel, for all the pleasant and fruitful talks that we had, and for her kindness as a "host" during my stays at Bell Labs.

My three years as a Ph.D. student at INRIA have been simply special, and this thanks to all the colleagues that I had the pleasure to meet, who made my working days instructive and at the same time really enjoyable. It is difficult to mention all of them without the risk of forgetting someone: a general big "thanks", then, to all the people who were part of the TEMICS and SIROCCO teams at any point since my arrival at INRIA in February 2011. Among them, I am particularly grateful to my two office mates, Dr. Mounira Ebdelli and Jérémy Aghaei Mazaheri, who shared with me nice everyday moments and tolerated my (often) nonsense talking. And a special mention also goes to the colleagues who alternated in the very international "Bureau F138" - my dear friend Dr. Mehmet Türkan, Dr. Raul Martínez Noriega, and lastly Júlio César Ferreira -, with whom I have established special connections that I hope will last a long time.

My life in Rennes was in general very pleasant, an ideal background for my work, and I would like to thank for this all the other friends at INRIA (particularly Dr. Bogdan Ludusan and Dr. Gylfi Thór Gudmundsson), the whole "Italian community", of which Dr. Roberto Cascella and Filippo Gaudenzi (both working at INRIA too) are good exponents, and many other "external" friends from France and many other countries, whose company I enjoyed much.

For the invaluable support over the years and for never making me feel alone, "grazie mille" also to my family, an essential reference point in my life.

Last but not least, I would like to deeply thank the person who supported me the most in the last critical months: many thanks, Deniz, for all your patience and affection towards me.

Contents

Introduction

Signal processing techniques to augment the visual quality of images and videos are nowadays particularly appealing. A first reason for this assertion is due to the technological progress that has raised the standards and the user expectations when enjoying multimedia contents. The past decade, in fact, has witnessed a revolution in large-size user-end display technology: consumer markets are currently flooded with television and other display systems - liquid crystal displays (LCDs), plasma display panels (PDPs), and many more -, which present very high-quality pictures with crystal-clear detail at high spatial and temporal resolutions. And the trend does not appear to be declining: only a few months ago, on the occasion of the annual Consumer Electronics Show (CES) held in January 2014, the company Seiki announced new 65-inch 4K TV models, which will be possible to buy at affordable prices. The recently adopted new compression standard HEVC, too, considered 4K videos for the tests, even if only on cropped areas.

However, despite the increasing interest towards them, high-quality contents are not always available to be displayed. Video source data are unfortunately often at a lower quality than the desired one, because of several possible causes: spatial and temporal down-sampling that may be necessary, noise degradation, high compression, etc. Moreover, the new sources of video contents, like the Internet or mobile devices, have generally a lower picture quality than conventional TV broadcasting. Lower-quality video sources pose a formidable obstacle to high-definition video display, and unfulfilled results are paradoxically even more accentuated by the new high-technology display systems. When we consider still images, things seem to be better. Modern cameras, even the handy and cheap ones, allow any user to easily produce breathtaking high-resolution photos. However, if we consider the past production, there is a tremendous amount of user-produced images collected in the years, that are valuable but may be affected by a poor quality. A need to improve the image quality can then be remarked also in this case.

Apart from the user experience, reasons for the need of augmenting the resolution of a video or an image can also be imposed by the particular application context considered. Many applications, e.g. video surveillance and remote sens-

ing, in fact, require the display of images at a considerable resolution, possibly for specific tasks like object recognition or zoom-in operations. Furthermore, compatibility requirements between the different video standards can sometimes lead to the necessity of implementing techniques of resolution up-conversion and de-interlacing.

Super-resolution (SR) specifically addresses the problem targeted so far, as it refers to a family of methods that aim at increasing the resolution, and thus the quality, of given images, to a greater extent than traditional image processing algorithms. Traditional methods include, among others, analytic interpolation methods, e.g. bilinear and bicubic interpolation, which compute the missing intermediate pixels in the enlarged high-resolution (HR) grid by averaging the original pixel of the low-resolution (LR) grid with fixed filters. Once the input image has been upscaled to HR via interpolation, image sharpening methods can be possibly applied. Sharpening methods aim at amplifying existing image details, by changing the spatial frequency amplitude spectrum of the image: in this way, provided that noise is not amplified too, existing high frequencies in the image are enhanced, thus producing a more pleasant and "richer" output image.

Differently than traditional methods (image interpolation, sharpening, etc.), the goal of SR is more ambitious: SR methods, in fact, aim at estimating missing high-resolution detail that is not present in the original image, by adding new plausible high frequencies. To pursue this goal, two main approaches to SR have been studied in the literature in the recent years: multi-frame and single-image super-resolution. Multi-frame SR methods, as the name suggests, rely on the presence of multiple frames, mutually misaligned and possibly originated by different geometric transformations, related to the same scene: these multiple frames are conveniently fused together to form a single HR output image. As a result, the formed image will contain an amount of detail that is not strictly present in any of the single input images, i.e. new information will be created. Single-image SR methods somehow represent an even bigger challenge, as we want here to create new high-frequency information from as little as one single input image. Like in image interpolation, we want to increase the spatial resolution of a LR input image, by making visible, in this case, new high-definition details.

Among single-image SR methods, an important category is represented by algorithms that make use of *machine learning* techniques. Although covering different meanings, machine learning can be generally referred to as that branch of artificial intelligence that concerns the construction and study of algorithms that can "learn from data". Its origins date back to several decades ago, but only in the nineties it definitely bloomed: since then, many powerful algorithms have been developed to solve different problems in a variety of scientific areas.

Interested in the SR approach to the task of increasing the resolution of an image, and intrigued by the effectiveness of machine learning techniques, dur-

ing this doctorate we mostly investigated the SR problem and the application to it of learning methods. In particular, we adopted a single-image "example-based" scheme, where a single LR image is upscaled by means of a dictionary of training examples. This dictionary of examples, which in this case consist in correspondences of LR and HR image *patches*, seen within the framework of machine learning, represents the data we have to learn from.

Example-based SR procedures are usually "patch-based" procedures: the input image itself is partitioned into patches, and, thanks to the correspondences stored in the dictionary, from a single LR input patch a single HR output patch is estimated via learning methods. The whole set of estimated HR patches is then re-assembled to finally build the super-resolved image. Starting from this general procedure, we investigated several algorithmic aspects of it, e.g. how to build the dictionary of patches or how to design the proper patch estimation procedure, by taking into account, from time to time, different targets (low complexity, maximization of the output quality, theoretical assessment, etc.). As for the dictionary, in particular, in example-based SR we can mainly have two different kinds of dictionary: an external one, when the training patches are extracted from external training images, or an internal one, when the patches are derived from the input image sequence itself. Broadly speaking, internal dictionaries allow to reach the best quality outcomes at the price of a higher computational cost, whereas external dictionaries lead to faster but less performing methods.

In this manuscript, we present novel single-image SR procedures for both kinds of dictionaries, i.e. external and internal, thus coming up with original solutions and competitive results w.r.t. state-of-the-art methods. We also extend our designed algorithm to the video case, i.e. when the task is to augment the resolution of a whole video sequence. This let us to already reach interesting results and to open the door to future work.

Thesis outline

The rest of this manuscript is structured as follows. We start with Chapter 1, where we give a general overview of super-resolution, by classifying it into multi-frame and single-image methods and also going deeper into the classification, and discuss the relevant work. Chapter 2 and Chapter 3 present our contributions to single-image SR, by describing novel algorithms employing, respectively, external and internal dictionaries. In particular, the external-dictionary methods presented in Chapter 2 are the result of several elements that brought to the formulation of three novel algorithms, described in distinct publications. The extension to the video case is presented in Chapter 4, where we consider two difference scenarios (according to whether or not some periodic HR frames are available). Different SR procedures (the single-image methods described in the

previous chapters, conveniently adapted to the video case, as well as newly designed procedures) are compared, and an analysis in the video compression context is also carried out. In Chapter 5 we then provide a compendium of some machine learning techniques that we have been studying during this doctorate, with a particular emphasis of their application within the SR techniques considered. Finally, we end the thesis by summarizing our accomplishments, drawing conclusions from them and discussing about future directions.

Chapter 1

Super-resolution: problem definition and review of the state of the art

In this chapter we present the fundamentals of super-resolution, by providing a general definition and classifying it, as conventionally done, into two main categories of algorithms: multi-frame methods and single-image methods. For these two families of algorithms, principles and main state-of-the-art methods are described, respectively in sections 1.1.1 and 1.1.2. Within single-image super-resolution methods, a particular family of methods that takes the name of example-based super-resolution, on which most of the work of this thesis focuses, is then further analyzed in Section 1.2.

1.1 What is super-resolution (SR)

Super-resolution, also spelled as super resolution and superresolution, is a term for a set of methods whose aim is to increase the *spatial resolution* of videos or images. Terms such as "upscale" and "upsize" also describe an increase of resolution in either image processing or video editing. Typical domains of application of super-resolution include medical imaging [1], remote sensing [2, 3], target detection and recognition [3, 4], radar imaging [5], forensic science [6], and surveillance systems [7].

In general terms, by image resolution it is meant the amount of detail an image holds. The resolution of an image can be described in many different ways, according to the particular aspect taken into account; thus, we can speak about spatial resolution, temporal resolution, spectral resolution, optical resolution, etc. In particular, spatial resolution refers to the level of visual details discernible in an image. In other words, it quantifies how close two lines in an image can be

to each other and still be visibly resolved. This definition applies to both digital and analogue camera systems and images. As for digital images, where the pixel is the base unit used, there is a connection between spatial resolution and total number of pixels. The number of effective pixels that an image sensor or digital camera has is the count of elementary pixel sensors that contribute to the final image: the higher this number will be, the higher the resolution will be and the clearer an object in the image will appear. In effect, spatial resolution can be referred to the number of "independent" pixel values per unit length. In practice, we often take into account simply the total counts of pixels, horizontally and vertically, which define what is more precisely called *pixel resolution*, and serve as upper bounds on spatial resolution (unused or light-shielded pixels around the edges might be included). In this case, the convention is to describe the pixel resolution with a set of two positive integer numbers, where the first number is the number of pixel columns (width) and the second is the number of pixel rows (height), e.g 1920×1080. In this manuscript, which only relates to digital images, the word "resolution" must be intended as a spatial resolution, unless otherwise specified, and the pixel-wise definition is used.

Super-resolution (SR) is not a marketing buzzword, but it is a proper mathematical term used by scientists. The first work on this topic was published in 1984 [8] and the term "super-resolution" itself appeared at around 1990 [9].

SR algorithms can be categorized according to the number of input images and output images involved in the process. When a single high-resolution (HR) image is produced from a single degraded low-resolution (LR) image, we refer to *single-image single-output (SISO) super-resolution*. Possible applications of SISO super-resolution relate to the possibility of achieving resolution enhancements, e.g. to improve object recognition performance and enable zoom-in capabilities. Other SR algorithms deal with the integration of multiple LR frames to estimate a unique HR image: in this case we speak about *multiple-input single-output (MISO) super-resolution*. An example application area is in license plate recognition from a video stream to increase the alphanumeric recognition rates. A recent focus on SR research relates to algorithms which aim at reconstructing a set of HR frames from an equivalent set of LR frames. This approach takes the name of *multiple-input single-output (MIMO) super-resolution*, also known as *video-to-video* SR. A typical application of these algorithms can be for example the quality enhancement of a video sequence captured by surveillance cameras. Table 1.1 sums up the characteristics of the three SR categories described (SISO, MISO, and MISO), by mentioning for each of them possible application domains.

The first two categories of SR algorithms (SISO and MISO) give rise to two different families of SR methods, each one recalling different kinds of procedures: particularly, for SISO super-resolution we speak about *single-image* SR methods (a single LR image is used as input to estimate an underlying HR image), whereas,

SR category	No. inputs	No. outputs	Applications
SISO	One	One	Image restoration algorithms, Object recognition
MISO	Several	One	Astronomical imaging, Medical imaging, Text recognition
MIMO	Several	Several	Video surveillance, Video enhancement

Table 1.1: Categories of SR algorithms - Differentiation between the three categories of SR algorithms (SISO, MISO, and MIMO) and some possible applications for each of them.

in the case of MISO super-resolution, we speak about *multi-frame* SR methods (the information contained within multiple under-sampled LR frames is fused to generate a single HR image). Several algorithms have been developed in the recent years for the two families of methods mentioned. MIMO super-resolution, instead, represents an emerging application, where procedures are not consolidated yet and often follow a sort of mixed approach, by borrowing elements either from SISO and MISO super-resolution algorithms. Thus, in terms of actual methodologies, we can distinguish two main families: single-image SR and multi-frame SR. Figure 1.1 presents a taxonomic diagram of SR according to the classifications made and the relations between the different categories of SR algorithms.

The principles behind multi-frame and single-image SR methods are deeply different. The former uses information from several different images to create one single upsized image. Algorithms try to extract and combine "real" details from every available frame. Single-image SR, instead, refers to more sophisticated methods (the problem is more difficult and necessarily ill-posed), which, starting from as little as one single input image, attempt at artificially synthesizing new details. Next two subsections will review these two families of SR algorithms, by presenting, for each of them, the main concepts and techniques involved.

1.1.1 Multi-frame SR methods

The basic premise for increasing the spatial resolution in multi-frame SR techniques is the availability of multiple LR images captured from the same scene. Multi-frame SR methods work effectively when several LR images contain slightly different perspectives of the scene to be super-resolved, i.e. when they represent different "looks" at the same scene. In this case, each image is seen as a degraded version of an underlying HR image to be estimated, where the degradation processes can include blurring, geometrical transformations, and down-sampling. If

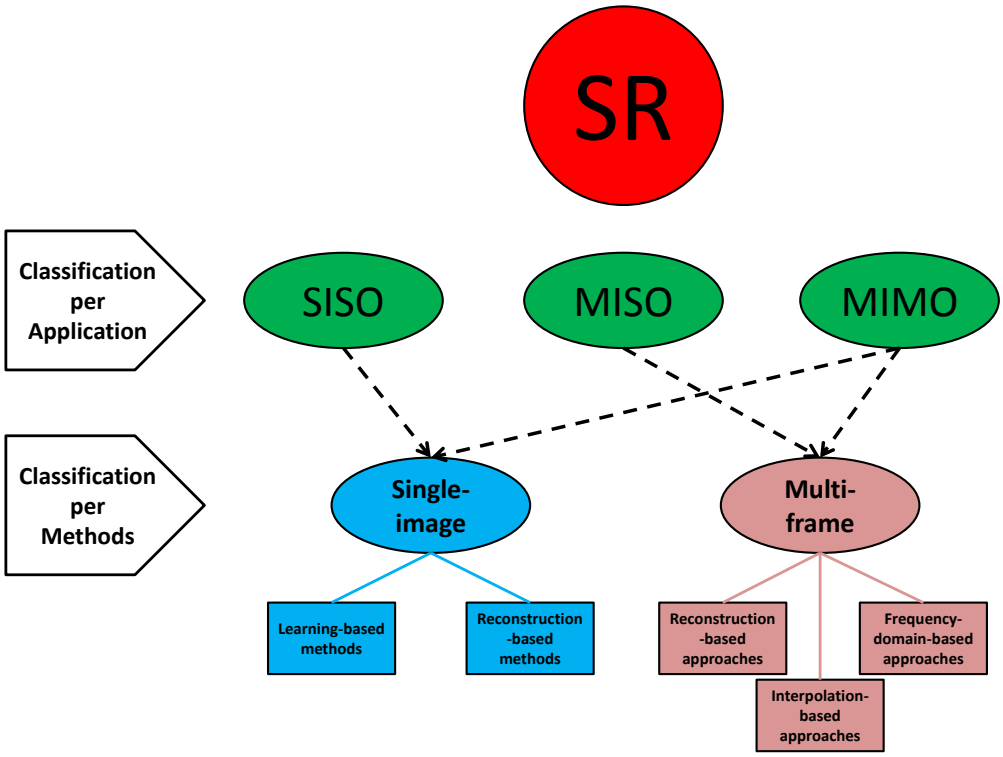


Figure 1.1: Taxonomic diagram of super-resolution - Classification of super-resolution algorithms from the point of view of applications (we distinguish between SISO, MISO, and MIMO super-resolution), and according to the different methodologies (in this case, we have single-image and multi-frame SR methods with their related sub-categories).

the geometrical transformations consist in simple shifts by integer units, then each image presents the same content and there is no extra information that can be exploited to reconstruct a common HR image. The best case for these methods to work is then when the LR images have different subpixel shifts from each other and thus they actually bring different information (each image cannot be obtained from the others).

The SR problem is usually modeled as an inverse problem, where the source information (the HR image) has to be estimated from the observed data (the LR frames). Solving an inverse problem generally requires first the construction of a forward *observation model*. Most imaging devices can be described as a camera lens and aperture that produce blurred images of the scene contaminated by some additional noise. Then, given a HR image with a total number of N pixels (the image can be expressed as a vector $\mathbf{x} \in \mathbb{R}^N$), the observation model that describes the acquisition of a set of p LR frames $\mathbf{y}_k \in \mathbb{R}^M$ can be expressed as:

$$\mathbf{y}_k = \mathbf{D}\mathbf{B}_k\mathbf{W}_k\mathbf{x} + \mathbf{n}_k \quad \text{for} \quad 1 \leq k \leq p \tag{1.1}$$

where $\mathbf{W}_k$ is an $N \times N$ warp matrix that maps the HR image coordinates onto a new system of distorted coordinates (due to translations, rotations, etc.), $\mathbf{B}_k$ is an $N \times N$ blur matrix modeling various phenomena (e.g. the optical system and the motion during the acquisition process), D is an $M \times N$ decimation matrix, and $\mathbf{n}_k$ represents an ordered noise vector. Therefore, the observed LR images result from warping, blurring, and subsampling operations performed on the same HR image x, which is implicitly considered constant during the whole acquisition process. If the subsampling factor considered is s (the LR images turn to be decimated by a factor of s in each spatial direction), then the dimensional relation that stands between the HR image vector x and the LR image vectors $\mathbf{y}_k$ is $N = s^2M$.

Multi-frame SR methods aim at reversing the observation model (1.1). In order to pursue this goal, many different methods have been proposed in the last decades. These methods, due to the difficulties in estimating the many parameters involved (e.g. a registration operation is often needed), are shown to be able to produce fairly good upscalings only for small scale factors (typically for factors smaller than 2). Moreover, the assumption for which several different frames referring to the same scene are available can be sometimes difficult to meet.

Broadly speaking, multi-frame SR algorithms can be classified according to three main approaches followed:

1. Interpolation-based approach
2. Frequency-domain-based approach
3. Regularization-based approach

In the following sections we revise the three approaches above mentioned, by presenting some of the most popular methods. Further information can be found in several review papers (e.g. [10, 11, 12, 13]), which have good overviews on multi-frame SR problems and the main methods employed.

1.1.1.1 Interpolation-based approach

Interpolation-based SR algorithms construct a HR image by projecting all the acquired LR images to a reference image. All the information available from each image is then fused together, due to the fact that each LR image provides an amount of additional information about the scene, and finally a deblurring process is applied to the obtained image. This procedure is depicted in Figure 1.2. Interpolation-based SR approaches usually consist of the following three stages:

- A *registration* stage for aligning the LR input images,

- An *interpolation* stage for producing a higher-resolution image, and
- A *deblurring* stage for enhancing the reconstructed HR image.

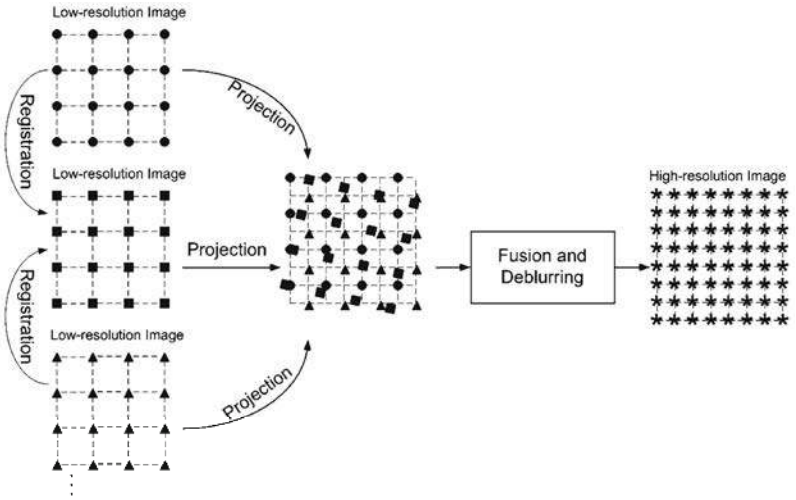


Figure 1.2: Scheme of the interpolation-based approach to multi-frame SR - The approach consists of three steps: registration, interpolation, and deblurring.

The interpolation stage plays a key role in this framework. There are various ways to perform interpolation: the simplest interpolation algorithm is the nearest neighbor algorithm, where each unknown pixel is assigned an intensity value that is same as its neighboring pixels. However, this method tends to produce images with a blocky appearance. Other traditional interpolation methods include bilinear and bicubic interpolation.

Ur and Gross [14] performed a *nonuniform interpolation* of a set of spatially shifted LR images, by utilizing the generalized multichannel sampling theorem of Papoulis [15]. The interpolation is followed by a deblurring process, and the relative shifts are assumed to be known precisely. The advantage of this approach is that it has low computational load, which makes it particularly suitable for real-time applications. However, the optimality of the entire reconstruction process is not guaranteed, since the registration errors are not taken into account.

Another nonuniform interpolation technique was implemented by Bose and Ahuja [16], who considered the error generated by inaccurate registration in the system matrix $\mathbf{W}_k$. In particular, they used the *moving least squares* method to estimate the intensity value at each pixel position of the HR image via a polynomial approximation using the pixels in a defined neighborhood of the pixel position under consideration. Furthermore, the coefficients and the order of the polynomial approximation are adaptively adjusted for each pixel position.

The three steps of the interpolation-based SR approach can also be conducted iteratively. In this regard, Irani and Peleg [9] proposed an iterative back-projection (IBP) algorithm, where the HR image is estimated by iteratively projecting the difference between the observed LR images and the simulated LR images. However, this method might not produce an unique solution due to the ill-posed nature of the SR problem. We will discuss further about the IBP algorithm in the next chapters, by employing it in the context of single-image SR.

Another iterative approach to multi-frame SR is given by the *Projection onto Convex Sets* (POCS) method, which was proposed, among others, by Patti *et al.* [17]. It is a set-theoretic algorithm that aims at producing an HR image that is consistent with the information arising from the observed LR images and a prior image model. According to the method of POCS, incorporating a priori knowledge into the solution can be interpreted as restricting the solution to be a member of a closed convex set C_i, which is defined as a set of vectors which satisfy a certain property. Given different properties for the HR target image, we then have several constraint sets: the intersection of these sets represents the space of permissible solutions. If the constraint sets have a nonempty intersection, then a solution that belongs to the intersection set $C_s = \bigcap_{i=1}^{m} C_i$, which is also a convex set, can be found by alternating projections onto these convex set. In fact, any solution in the intersect set is consistent with the a priori constraints and therefore a feasible solution. The method of POCS consists then in projecting an initial estimate of the unknown HR $\mathbf{x}^0$ onto these constraint sets, according to the following recursion:

$$\mathbf{x}^{n+1} = \mathbf{P}_m \mathbf{P}_{m-1} \cdots \mathbf{P}_2 \mathbf{P}_1 \mathbf{x}^n \tag{1.2}$$

where $\mathbf{x}^0$ is an arbitrary starting point, and P_i is the projection operator which projects an arbitrary signal $\mathbf{x}$ onto the closed convex set C_i. With (1.2), a fairly good solution to the problem can be obtained: although not trivial, this is, in general, much easier than finding P_s, i.e. the unique projector, in one step. The method of POCS is easy to implement; however, it does not guarantee the uniqueness of the solution. Furthermore, the computational cost of this algorithm is very high.

1.1.1.2 Frequency-domain-based approach

A major class of multi-frames SR methods utilizes a frequency domain formulation of the SR problem. The main principle is that clues about high frequencies are spread across the multiple LR images in form of aliased spectral frequencies.

The first frequency-domain SR method can be credited to Tsai and Huang [8], who considered SR reconstruction from noise-free LR images. They proposed

to first transform the LR image data into the Discrete Fourier Transform (DFT) domain, and then combine them according to the relationship between the aliased DFT coefficients of the observed LR images. The approach is based on the following principles:

1. The shifting property of the Fourier transform,
2. The aliasing relationship between the continuous Fourier transform (CFT) and the DFT of observed LR images, and
3. The assumption that an original HR image is band-limited.

These properties make it possible to formulate an equation relating the aliased DFT coefficients of the observed LR images to a sample of the CFT of an unknown HR image.

Rhee and Kang [18] exploited the Discrete Cosine Transform (DCT), instead of DFT, in order to reduce memory requirements and the computational costs. Woods *et al.* [19], instead, presented an iterative expectation maximization (EM) algorithm [20] for simultaneously performing the registration, blind deconvolution, and interpolation operations.

The frequency-domain-based SR approach has a number of advantages. The first advantage is its theoretical simplicity: the relationship between the LR input images and the HR image is clearly demonstrated. Thus, frequency-domain-based SR approaches represent an intuitive way to enhance the details of the images by extrapolating the high-frequency information presented in the LR images. Secondly, these approaches have low computational complexity, by also being suitable for parallel implementations. However, frequency-domain-based SR methods are insufficient to handle real-world applications, since they require that there only exists a global displacement between the observed images and the linear space-invariant blur during the image acquisition process (i.e. $\mathbf{B}_k$ is supposed to be the same for all the LR images).

Recently, many researchers have begun to investigate the use of the wavelet transform for addressing the SR problem to recover the detailed information (usually the high-frequency information) that is lost or degraded during the image acquisition process. This is motivated by the fact that the wavelet transform provides a powerful and efficient multi-scale representation of the image for recovering the high-frequency information [21]. These approaches typically treat the observed LR images as the low-pass filtered sub-bands of the unknown wavelet-transformed HR image. The aim is to estimate the finer scale sub-band coefficients, followed by applying the inverse wavelet transform, to produce the HR image. To be more specific, let us take the 2×2 SR computation as an example. The LR images are viewed as the representation of wavelet coefficients after N levels of decomposition. Then, the HR image can be produced by estimating the $(N+1)$-*th* scale wavelet coefficients, followed by applying the inverse wavelet

decomposition. In [22], El-Khamy *et al.* proposed to first register multiple LR images in the wavelet domain, then fuse the registered LR wavelet coefficients to obtain a single image, and finally perform interpolation to get a higher-resolution image. Ji and Fermüller [23, 24] proposed a robust wavelet SR approach to handle the error incurred in both the registration computation and the blur identification computation. Chappalli and Bose incorporated in [25], instead, a denoising stage into the conventional wavelet-domain SR framework to develop a simultaneous denoising and SR reconstruction approach.

1.1.1.3 Regularization-based approach

Motivated by the fact that the SR computation is, in essence, an ill-posed inverse problem, due to the insufficient number of LR images or ill-conditioned blur operators, numerous regularization-based SR algorithms have been developed to "stabilize" the inversion operation, by decreasing the number of possible solutions. The basic idea of these regularization-based SR approaches is to use the regularization strategy to incorporate some prior knowledge of the unknown HR image. The related methods can be broadly classified into two categories:

- Deterministic regularization approaches, and
- Stochastic regularization approaches.

Deterministic approaches rely on the fact that, with estimates of the registration parameters, the observation model in (1.1) can be completely specified. The deterministic regularization SR approach then solves the inverse problem in (1.1), by exploiting some prior information about the solution, which can be used to make the problem well posed. A typical method adopted refers to the solution of a constrained least squares (CLS) problem. The CLS problem can be formulated, e.g., by choosing an $\mathbf{x}$ to minimize a Lagrangian [26], according to the following formulation:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \{ \sum_{k=1}^{p} \|\mathbf{y}_k - \mathbf{A}_k \mathbf{x}\|^2 + \alpha \|\mathbf{C}\mathbf{x}\|^2 \} \ , \tag{1.3}$$

where $\mathbf{A}_k$ is the global matrix that takes into account all the degradation operations on the frame $\mathbf{y}_k$, α is the Lagrange multiplier, and $\mathbf{C}$ is generally a high-pass filter. In (1.3), a priori knowledge concerning a desirable solution is represented by a smoothness constraint, suggesting that most images are naturally smooth and therefore it is appropriate to minimize the amount of high-pass energy in the restored image. The Lagrange multiplier α controls the trade-off between fidelity to the data and smoothness of the solution. The cost functional in (1.3) is convex and differentiable with the use of a quadratic regularization term. Therefore, we can find a unique estimate $\hat{\mathbf{x}}$ which minimizes it. One basic deterministic iterative

technique considers solving

$$\left(\sum_{k=1}^{p}\mathbf{A}_k^{\mathrm{T}}\mathbf{A}_k+\alpha\mathbf{C}^{\mathrm{T}}\mathbf{C}\right)\hat{\mathbf{x}}=\sum_{k=1}^{p}\mathbf{A}_k^{\mathrm{T}}\mathbf{y}_k\ , \tag{1.4}$$

which leads to the following iteration for $\hat{\mathbf{x}}$:

$$\hat{\mathbf{x}}^{n+1}=\hat{\mathbf{x}}^{n}+\beta\left(\sum_{k=1}^{p}\mathbf{A}_k^{\mathrm{T}}(\mathbf{y}_k-\mathbf{A}_k\hat{\mathbf{x}}^{n})-\alpha\mathbf{C}^{\mathrm{T}}\mathbf{C}\hat{\mathbf{x}}^{n}\right)\ , \tag{1.5}$$

where β represents the convergence parameter.

Stochastic regularization methods typically adopt a Bayesian approach, according to which the information that can be extracted from the observations (i.e. the LR images) about the unknown signal (i.e. the HR image) is contained in the probability distribution of the unknown. These methods rely then on the possibility that the a posteriori probably density function (PDF) of the original HR image can somehow be established, by exploiting the information provided by both the observed LR images and the prior knowledge of the HR image.

One of the most popular Bayesian-based SR approaches is the maximum a posteriori (MAP) estimation approach [27, 28, 29, 30, 31, 32]. The MAP estimator of $\mathbf{x}$ maximizes the a posteriori PDF $P(\mathbf{x}|\mathbf{y}_k)$ with respect to $\mathbf{x}$, i.e.:

$$\hat{\mathbf{x}}=\arg\max_{\mathbf{x}}P(\mathbf{x}|\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_p)\ . \tag{1.6}$$

Taking the logarithmic function and applying the Bayes' theorem to the conditional probability, the MAP optimization can be expressed as:

$$\hat{\mathbf{x}}=\arg\max_{\mathbf{x}}\left\{\ln P(\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_p|\mathbf{x})+\ln P(\mathbf{x})\right\}\ . \tag{1.7}$$

Both the a priori image model $P(\mathbf{x})$ and the conditional density $P(\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_p|\mathbf{x})$ are defined by a priori knowledge concerning the HR image $\mathbf{x}$ and the statistical information of noise.

Another popular Bayesian-based approach is maximum likelihood (ML) estimation, which is a special case of MAP estimation with no prior term. ML estimation has also been applied to the SR reconstruction problem [33]. However, due to the ill-posed nature of SR inverse problems, MAP estimation is usually preferred to ML.

1.1.2 Single-image SR methods

Single-image SR is the problem of estimating an underlying HR image, given only one observed LR image. In this case, it is assumed that there is no access

to the imaging step so that the starting point is a given LR obtained according to some known or unknown conventional imaging process.

The generation process of the LR image from the original HR image that is usually considered can be written as

$$I_L = (I_H * b) \downarrow_s , \tag{1.8}$$

where I_L and I_H are respectively the LR and HR image, b is a blur kernel the original image is convoluted with, which is typically modeled as a Gaussian blur [34], and the expression $\downarrow_s$ denotes a downsampling operation by a scale factor of s. The LR image in then a blurred and down-sampled version of the original HR image.

Single-image SR aims at constructing the HR output image from as little as a single LR input image. The problem stated is an inherently ill-posed problem, as there can be several HR images generating the same LR image. Single-image SR is deeply connected with traditional "analytical" interpolation, as they share the same goal. Traditional interpolation methods (e.g. linear, bicubic, and cubic spline interpolation [35]), by computing the missing pixels in the HR grid as averages of known pixels, implicitly impose a "smoothness" prior. However, natural images often present strong discontinuities, such as edges and corners, and thus the smoothness prior results in producing ringing and blurring artifacts in the output image. The goal of SR is thus to achieve better results, by using more sophisticated statistical priors.

Single-image SR algorithms can be broadly classified into two main categories:

1. *learning-based methods*, which make use of machine learning techniques and often employ a dictionary generated from an image database;
2. *reconstruction-based methods*, which do not use a training set but rather define constraints for the target high-resolution image to improve the quality of the reconstruction.

Reconstruction-based single-image SR includes a variety of methods. Here, the prior information necessary to solve the single-image SR ill-posed problem is usually available in the explicit form of either a distribution or an energy functional defined on the image class. Several algorithms of this kind are edge-focused methods, i.e. they try to reconstruct image details by interpolating the LR input image while focusing on sharpening edges [36, 37, 38, 39, 40]. E.g. in the approach of Dai *et al.* [38], the edges of the input image are extracted, in order to enforce their continuity, and blended together with the interpolation result to yield to the the final super-resolved image. Similarly, in [40] Fattal proposes a method where edge statistics are used to reconstruct the missing high frequency information. Another approach relates to the attempt to solve the ill-posed problem of SR through regularization methods. E.g. the method in [41] adds to

the problem a total variation (TV) regularization term. Following the theory of compressed sensing, the authors in [42] propose instead a "compressive image super-resolution" framework, where they enforce the constraint that the HR image be sparse in the wavelet domain. Some methods proposed by Dong *et al.* [43, 44] follow a mixed approach: while using dictionaries of patches (that would induce to classify them as learning-based methods), they can still be considered belonging to the reconstruction-based family, as the HR image is computed by solving an optimization problem with several regularization terms. Without following either the edge-focused or the regularization-based approaches, Shan *et al.* instead propose in [45] a fast image upsampling method with a feedback-control scheme performing image deconvolution.

The distinctive feature of learning-based methods, as said, is the employment of machine learning techniques to locally estimate the HR details of the output image. Learning-based algorithms can consist in pixel-based procedures, where each value in the HR output image is singularly inferred via statistical learning [46, 47], or *patch-based procedures*, where the HR estimation is performed thanks to a dictionary of correspondences of LR and HR patches (i.e. squared blocks of image pixels). The dictionary generated in this way (relating LR patches to HR patches) is then applied to the given LR image to recover its "most-likely" HR version. Note that this estimated HR version depends on the goodness of the dictionary; therefore, the reconstruction of true (unknown) details is not guaranteed. For this reason these methods are also referred to as "image hallucination" methods.

Learning-based single-image SR that makes use of patches is also referred to as *example-based* SR [48]. In the upscaling procedure the LR input image itself is divided into patches, and for each LR input patch a single HR output patch is reconstructed, by observing the "examples" contained in the dictionary. In the original example-based algorithm of Freeman *et al.*, for example, the LR input image is subdivided into the overlapping patches, to form a Markov Random Field (MRF) framework. By searching for nearest neighbors in a LR dictionary, a certain number of corresponding HR candidates is then retrieved. This results in a MRF with a number of HR candidate patches for each node. After associating a data fitting cost to each candidate and a continuity cost to the neighboring candidates, the MRF can be solved by using techniques such as belief propagation. One drawback of this scheme is its high computational cost, due to the complex solution and to the necessity of having large dictionaries including a large variety of image patches. As for the first point, Freeman *et al.* adopt in [48] a sub-optimal one-step approach to solve the global optimization problem. To directly or indirectly face the issue of the dictionary size, instead, many example-based SR algorithms consisting in different procedures have been proposed in the literature in the recent years. *Neighbor embedding SR* methods [49, 50, 51] are

characterized by the selection of several LR candidate patches in the dictionary via nearest neighbor search; the HR output patches are reconstructed by combining the HR versions of these selected candidates. In this way, since patch combinations are enabled (i.e. the patch subspace is interpolated thus yielding more patch patterns), the number of image patch exemplars needed can be ideally lowered, while maintaining the same "expressive power" of the dictionary. Another family of example-based SR method, *sparse coding SR* [52, 53, 54], is based on the concept of sparse representations: the weights of each patch combination are in this case computed by sparsely coding the related LR input patch with the patches in the dictionary. Dictionary learning methods can then be used to train a more compact dictionary (i.e. resulting with a lower number of patch pairs), particularly suitable for sparse representations. The method presented in [55] somehow bridges the neighbor embedding and the sparse coding approaches, by proposing a "sparse neighbor embedding" algorithm.

In the next section we provide an ample overview of example-based SR, by systematically presenting the main concepts and classifying the techniques used.

1.2 Example-based super-resolution

Example-based single-image SR aims at reversing the image generation model (1.8), by means of a dictionary of image *examples* that maps locally the relation between an HR image and its LR counterpart, the latter obtained with the model (1.8). For general upscaling purposes, the examples used are typically in the form of patches, i.e. squared blocks of pixels (e.g. 3×3 or 5×5 blocks). The dictionary is then a collection of patches, which, two by two, form pairs. A pair specifically consists of a LR patch and its HR version with enriched high frequency details.

Example-based SR algorithms comprise two phases:

1. A *training phase*, where the above-mentioned dictionary of patches is built;
2. The proper *super-resolution phase*, that uses the dictionary created to upscale the input image.

As for the training phase, in Section 1.2.1 we discuss how to build a dictionary of patches. The dictionary can be of two kinds: an external dictionary, built from a set of external training images, and an "internal" one, built without using any other image than the LR input image itself. This latter case exploits the so called "self-similarity" property, typical of natural images, according to which image structures tend to repeat within and across different image scales: therefore, patch correspondences can be found in the input image itself and possibly scaled versions of it. To learn these patch correspondences, that specifically take the name of *self-examples*, we can have one-step schemes [56, 57] or schemes based on a pyramid of recursively scaled images starting from the LR input image

[58, 59, 60]. Clearly, the advantage of having an external dictionary instead of an internal one lies in the fact that it is built in advance, while the internal one is generated online and updated at each run of the algorithm. However, external dictionaries have a considerable drawback: they are fixed and thus non-adapted to the input image. The study conducted in [61] also confirms the benefit of using internal statistics in patch-based image processing algorithms (Figure 5 in [61] shows that the gap in "expressiveness" between internal and external dictionaries, intended as the average error of several patch approximations, increases for higher gradient contents of the patches).

As for the super-resolution phase, in example-based algorithms the patch is also the reconstruction unit used in the upscaling procedure. In fact, the LR input image is partitioned into patches; for each single LR input patch, then, by using the LR-HR patch correspondences in the dictionary, a HR output patch is constructed. The HR output image is finally built by re-assembling all the reconstructed HR patches. Figure 1.3 shows in a simple manner the operating diagram of an example-based SR algorithm, according to the principle described above.

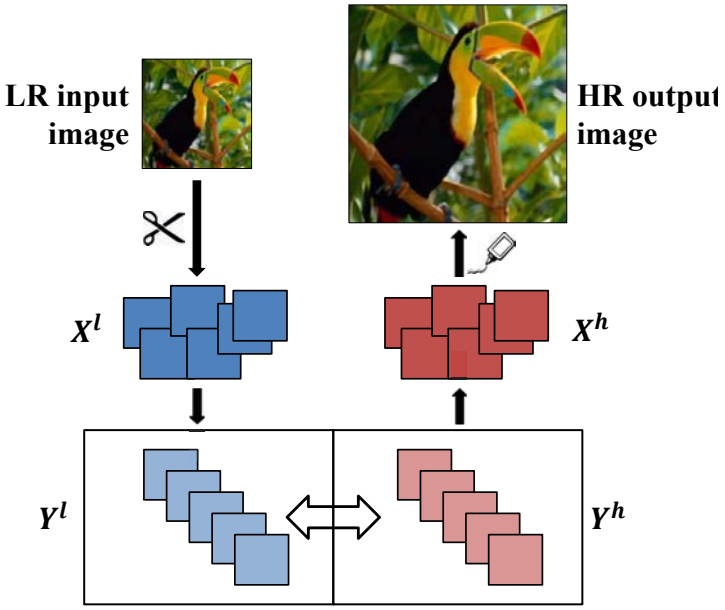


Figure 1.3: General scheme of the example-based SR procedure - The input image is partitioned into LR patches, a dictionary of patch correspondences is exploited, and new HR patches are generated and subsequently re-assembled to create the super-resolved output image.

In Section 1.2.2 we revise the two main patch reconstruction approaches adopted in SR procedures: local learning approaches, based on nearest neighbor

searches, and sparse coding approaches, which rely on some sparse assumption.

Hereinafter in this manuscript we will use the following notation to indicate the different kinds of patches. $\mathcal{X}^l = \{\mathbf{x}_i^l\}_{i=1}^{N_x}$ will denote the set of overlapping LR patches into which the LR input image is partitioned; similarly, $\mathcal{X}^h = \{\mathbf{x}_i^h\}_{i=1}^{N_x}$ will denote the set of reconstructed HR patches, that will form the HR output image. Each patch is expressed in vector form, i.e. its pixels, or certain features computed on them, are concatenated to form a unique vector. As for the dictionary, $\mathcal{Y}^l = \{\mathbf{y}_i^l\}_{i=1}^{N_y}$ and $\mathcal{Y}^h = \{\mathbf{y}_i^h\}_{i=1}^{N_y}$ will be respectively the sets of LR and HR example patches. In each patch reconstruction, then, the goal is to predict an HR output patch $\mathbf{x}_i^h$, given the related LR input patch $\mathbf{x}_i^l$, and the two coupled sets of patches, $\mathcal{Y}^l$ and $\mathcal{Y}^h$, that form the dictionary. Table 1.2 recaps the notation used in this manuscript for the different kinds of patches.

$\mathcal{X}^l = \{\mathbf{x}_i^l\}_{i=1}^{N_x}$	LR patch vectors in the LR input image
$\mathcal{X}^h = \{\mathbf{x}_i^h\}_{i=1}^{N_x}$	HR patch vectors in the HR output image
$\mathcal{Y}^l = \{\mathbf{y}_i^l\}_{i=1}^{N_y}$	LR patch vectors in the dictionary
$\mathcal{Y}^h = \{\mathbf{y}_i^h\}_{i=1}^{N_y}$	HR patch vectors in the dictionary

Table 1.2: Notation used for the different kinds of patch vectors - The letters "x" and "y" indicate the two possible sources of patches (test image or dictionary, respectively); LR and HR patches are present in both cases.

1.2.1 Types of dictionary

As said, when building a dictionary of patches for example-based SR, we have two possible choices:

- An external dictionary, and
- An internal dictionary.

In order to construct an external dictionary, several high-definition natural images are taken as training data. For each training image J_H, we generate an LR counterpart J_L by following the same image generation process assumed for SR (1.8), i.e.

$$J_L = (J_H * b)\downarrow_s \quad . \tag{1.9}$$

LR and HR patches are extracted respectively from J_L and J_H, or from processed versions of them. Two patches are considered to form a pair when coming from corresponding locations in a pair of training images (the HR original image and its LR counterpart). The dictionary thus formed can be used directly as a training set in the second phase of the algorithm (super-resolution phase).

In example-based SR, when performing the training phase, we speak about an internal learning if, instead of making use of external training images, we derive the patches, directly or indirectly, from the input image itself. Local image structures, that can be captured in the form of patches, tend to recur across different scales of an image, especially for small scale factors. We can then use the input image itself, conveniently up-sampled or down-sampled into one or several differently re-scaled versions, and use pairs of these images to learn correspondences of LR and HR patches, that will constitute our internal dictionary. We call the patches learned in this way *self-examples*. In this respect, there are two main kinds of learning schemes described in the literature: one-step schemes like in [56, 57], and schemes involving the construction of a pyramid of recursively scaled images [58, 59, 60].

One-step schemes are meant to reduce as much as possible the size of the internal dictionary, i.e. the number of self-examples to be examined at each patch reconstruction, and thus the computational time of the SR algorithm. In fact, from the input image only one pair of training images is constructed and thus taken into account for the construction of the dictionary. This approach is motivated by the fact that the most relevant patches correspondences can be found when the re-scale factor employed is rather small. Only one rescaling is then sufficient to obtain a good amount of self-examples. Let $\mathcal{D}$ denote an image downscaling operator, s.t. $\mathcal{D}(I) = (I) \downarrow_p$, where p is a conveniently chosen small scale factor; and let $\mathcal{U}$ denote the dual upscaling operator, s.t. $\mathcal{U}(I) = (I) \uparrow_p$. Let I_L still indicate the LR input image. In [56], for example, $J_L = \mathcal{U}(\mathcal{D}(I_L))$, which represents a low-pass filtered version of the LR input image I_L, is used as source for the LR patches, whereas the HR example patches are sampled from $J_H = I_L - J_L$, the complementary high-pass version. In [57], instead, we have $J_L = (I_L * b)$ and $J_H = \mathcal{U}(I_L)$: the LR examples patches are taken again from a low-pass filtered version of I_L, obtained by blurring the original image with a Gaussian blur kernel b, and the corresponding HR patches are taken from an upscaled version of I_L, which does not alter its frequency spectrum content.

The method described in [58] paved the way to several SR algorithms (e.g. [59, 60]), based on self-examples derived in an image pyramid. Differently from one-step learning methods, here several training images are constructed by recursively down-scaling the LR input image, thus forming a sort of overturned pyramid. Given a LR input patch, all the levels of this pyramid can be used to find similar self-examples, usually by performing a nearest neighbor search (NNS) (see Fig. 1.4).

To test the effective usefulness of constructing a full pyramid of images, we built a pyramid with 6 sub-levels (the original LR image is recursively down-scaled 6 times). For each LR input patch, then, the $K = 9$ most similar self-examples (the K nearest neighbors) have been searched throughout the whole pyramid, and

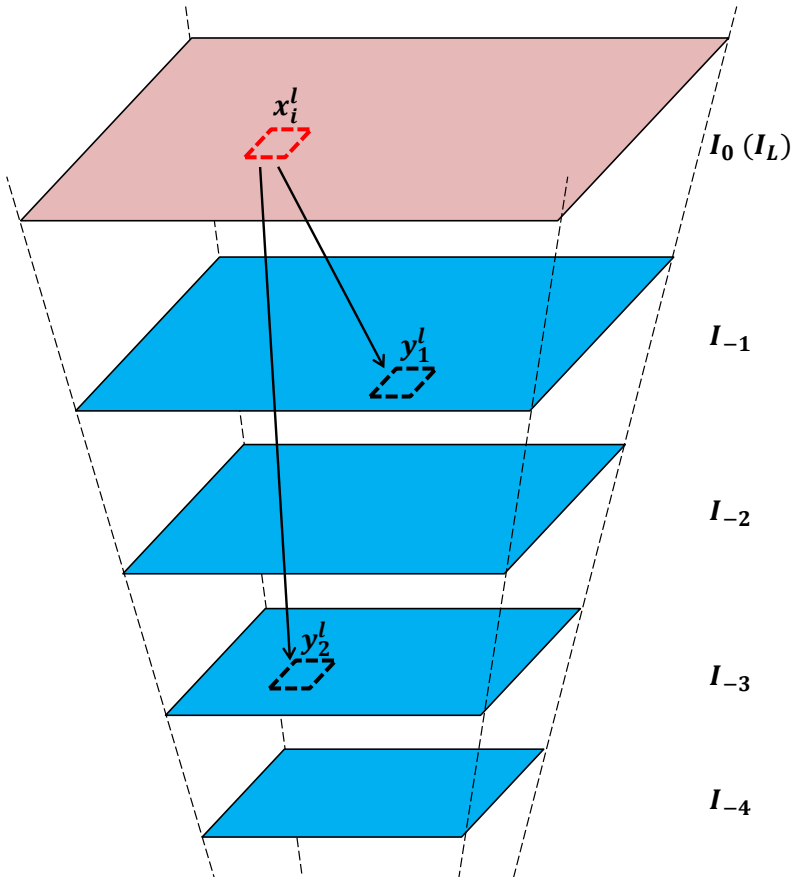


Figure 1.4: Pyramid of recursively scaled images - The pyramid consists in this case of 4 sub-levels, where the top level (I_0) is represented by the LR input image I_L. Given a LR input patch $\mathbf{x}_i^l$, a desired number of "neighbors" $(\mathbf{y}_1^l, \mathbf{y}_2^l, \dots)$ can be found at any level of the pyramid.

the histogram of all the selected neighbors has been drawn, where the histogram classes are the 6 sub-levels of the pyramid. Figure 1.5 presents the histograms of the selected neighbors for two different images.

As we can observe from Figure 1.5, it is clear that the first sub-level, i.e. the image obtained with only one rescaling operation, is the most relevant source of self-examples. Nevertheless, in both cases nearly 35 percent of the neighbors still come from the other sub-levels, which is not a negligible percentage.

1.2.2 Patch reconstruction methods

Once the dictionary of self-examples is built, i.e. we have the two dictionary sets $\mathcal{Y}^l$ and $\mathcal{Y}^h$ containing, respectively, LR and HR example patches, the proper SR upscaling procedure is ready to start. In this respect, example-based SR

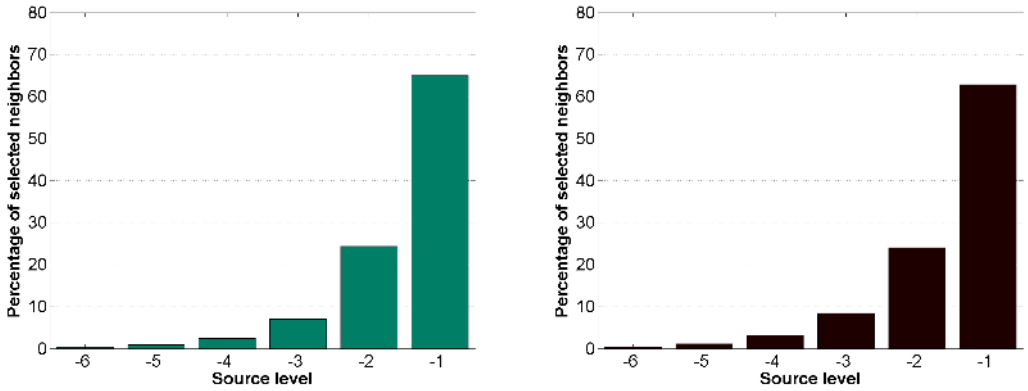


Figure 1.5: Percentage of selected neighbors for each sub-level of the pyramid - The simple test, performed on two different images, shows that, while the first sub-level being the most used one, other sub-levels have a certain importance too.

algorithms consist in patch-based procedures, where the HR output image is built by means of single patch reconstructions, as many as the number of LR patches the LR input image is partitioned into. The goal of each patch reconstruction is to predict a HR patch $\mathbf{x}_i^h$ from the related known LR patch $\mathbf{x}_i^l$, with the help of the dictionary sets.

We can broadly classify patch reconstruction methods for example-based SR into two categories:

- Local learning methods (see Section 1.2.2.1), and
- Sparse coding methods (see Section 1.2.2.2).

1.2.2.1 Local learning methods

Local learning methods are characterized by the fact that the estimations of the output patches are made by observing "locally" the dictionary. For each input patch $\mathbf{x}_i^l$, LR and HR *local* training sets are formed, by performing a nearest neighbor search (NNS). A desired number of neighbors of the LR input patch $\mathbf{x}_i^l$ is searched among the LR self-examples $\mathcal{Y}^l$, and consequently an equal number of HR neighbors is determined. Let Y_i^l indicate the matrix collecting, column by column, the selected LR neighbors, and let Y_i^h indicate the matrix of the corresponding HR neighbors. Thanks to the local sets Y_i^l and Y_i^h, the unknown HR patch $\mathbf{x}_i^h$ is then predicted. The whole *local learning* procedure can be summarized in 3 steps.

1. **Nearest Neighbor Search (NNS)**: The local training sets Y_i^l and Y_i^h are determined.

2. **Model generation**: A model $\mathcal{M}_i$ for the local reconstruction is computed. $\mathcal{M}_i$ generally depends on both the LR input patch and the local training sets: $\mathcal{M}_i = \mathcal{M}(\mathbf{x}_i^l, \mathbf{Y}_i^l, \mathbf{Y}_i^h)$.
3. **Prediction**: The model $\mathcal{M}_i$ is applied to actually predict the HR output patch.

Among local learning methods, we can distinguish two main reconstruction approach: neighbor embedding (NE) and direct mapping (DM). NE and DM differ in steps 2 and 3 of the local learning procedure above. In NE, the reconstruction model $\mathcal{M}_i$ consists of a vector of weights $\mathbf{w}_i \in \mathcal{R}^K$ (where K is the number of neighbors chosen), that identifies a linear combination of the LR neighbors $\mathbf{Y}_i^l$. The weights are computed w.r.t. the LR input patch and its neighbors ($\mathbf{w}_i = \mathcal{M}(\mathbf{x}_i^l, \mathbf{Y}_i^l)$). In [58], e.g., the single weight $w_i(j)$, related to the neighbor $\mathbf{y}_j^l$ found in the pyramid, is an exponential function of the distance between the latter and the LR input patch, according to the non-local means (NLM) model [62].

$$w_i(j) = \frac{1}{C} e^{-\frac{\|\mathbf{x}_i^l - \mathbf{y}_j^l\|_2^2}{t}} , \tag{1.10}$$

where t is a parameter to control the decaying speed and C is a normalizing constant to make the weights sum up to one.

In other NE methods, instead, the weights are meant to describe a linear combination that approximates the LR input patch (i.e. $\mathbf{x}_i^l \approx \mathbf{Y}_i^l \mathbf{w}_i$). In [49, 50, 51], e.g., the weights are computed as the result of a least squares problem with a sum-to-one constraint:

$$\mathbf{w}_i = \arg\min_{\mathbf{w}} \|\mathbf{x}_i^l - \mathbf{Y}_i^l \mathbf{w}\|^2 \quad \text{s.t.} \quad \mathbf{1}^T \mathbf{w} = 1 . \tag{1.11}$$

The formula (1.11) recalls the method used in Locally Linear Embedding (LLE) [63] to describe a high-dimensional point lying on a manifold through its neighbors, where the sum-to-one constraint is meant to make the weights even more representative of the local geometry of the manifold (it makes them invariant to translations of the data points).

In [64], instead, the sum-to-one constraint is replaced by a nonnegative condition, thus enabling only additive combinations of patches:

$$\mathbf{w}_i = \arg\min_{\mathbf{w}} \|\mathbf{x}_i^l - \mathbf{Y}_i^l \mathbf{w}\|^2 \quad \text{s.t.} \quad \mathbf{w} \geq 0 . \tag{1.12}$$

Once the vector of weights $\mathbf{w}_i$ is computed, the prediction step (Step 3) of the NE learning procedure consists in generating the HR output patch $\mathbf{x}_i^h$ as a linear combination of the HR neighbors $\mathbf{Y}_i^h$, by using the same weights:

$$\mathbf{x}_i^h \approx \mathbf{Y}_i^h \mathbf{w}_i . \tag{1.13}$$

Figure 1.6 depicts the scheme of the patch reconstruction procedure in the NE case. The model, i.e. the weights, is totally learned in the LR space, and then applied to the HR local training set to generate the HR output patch.

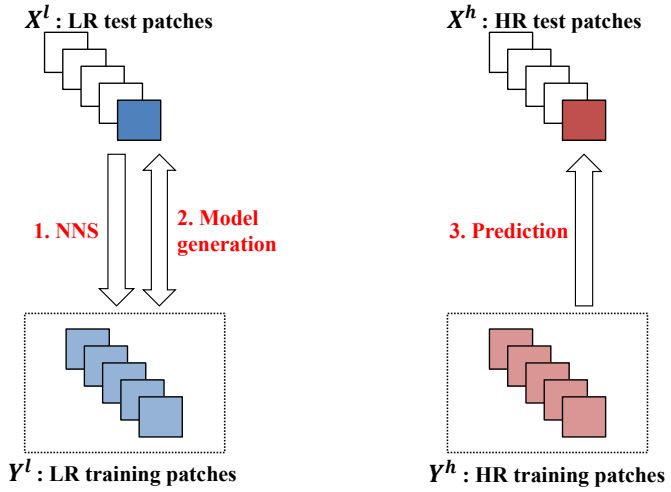


Figure 1.6: Scheme of the neighbor embedding (NE) reconstruction procedure - NE exploits the intra-space relationships, by learning a model among the LR patches and applying it on the HR patches.

In direct mapping (DM) methods [57, 65, 66], instead, the model is a regression function f_i that directly maps the LR input patch into the HR output patch. The function is learned by taking into account the two local training sets ($f_i = \mathcal{M}(\mathbf{Y}_i^l, \mathbf{Y}_i^h)$), by minimizing the empirical fitting error between all the pairs of examples, with possibly a regularization term:

$$f_i = \arg\min_{f \in \mathcal{H}} \sum_{j=1}^{K} \|\mathbf{y}_j^h - f(\mathbf{y}_j^l)\|_2^2 + \lambda \|f\|_{\mathcal{H}}^2 \,, \tag{1.14}$$

where $\mathcal{H}$ is a desired Hilbert function space and $\lambda \geq 0$ a regularization parameter. Examples similar to $\mathbf{x}_i^l$ and their corresponding HR versions are then used to learn a unique mapping function, which is afterwards simply applied to $\mathbf{x}_i^l$ to predict the HR output patch (Step 3):

$$\mathbf{x}_i^h = f_i(\mathbf{x}_i^l) \tag{1.15}$$

Figure 1.7 depicts the scheme of the patch reconstruction procedure also in the DM case.

1.2.2.2 Sparse coding methods

As another patch reconstruction method for example-based SR, sparse coding methods estimate the HR output patches via sparse representations [52, 53]. Here,

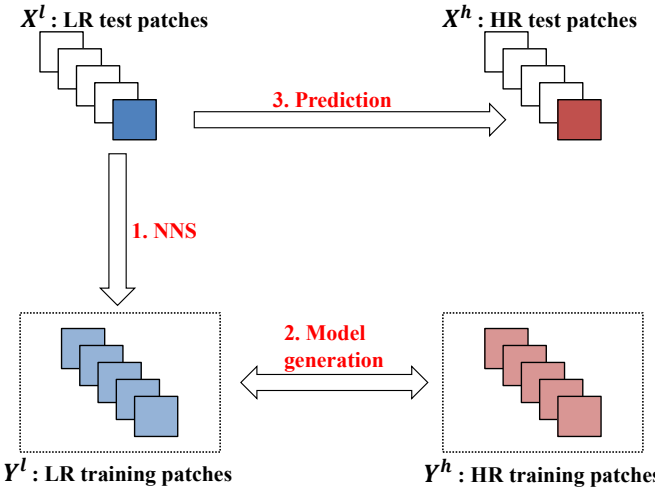


Figure 1.7: Scheme of the direct mapping (DM) reconstruction procedure - DM exploits the "horizontal" relationships between LR and HR patches, by attempting to learn the mapping between the two spaces.

the patch reconstruction is performed according to the principles of sparsity: for each LR input patch $\mathbf{x}_i^l$, we find a sparse vector α_i with respect to the LR dictionary $\mathcal{Y}^l$ (the length of α_i is equal to the number of patches in the dictionary); the elements of the HR dictionary $\mathcal{Y}^h$ will be combined according to the same coefficients to generate the HR output patch $\mathbf{x}_i^h$.

The patch reconstruction procedure of sparse coding methods (for a given LR input patch $\mathbf{x}_i^l$) consists then of two steps.

1. Find the sparse representation α_i to sparsely approximate the input patch (*coding step*)
$$\min \|\alpha_i\|_0 \quad \text{s.t.} \quad \|\mathbf{Y}^l \alpha_i - \mathbf{x}_i^l\| \leq \epsilon \; . \tag{1.16}$$

2. Apply the same coefficients to sparsely generate the corresponding HR output patch (*synthesizing step*)
$$\mathbf{x}_i^h = \mathbf{Y}^h \alpha_i \; , \tag{1.17}$$
where $\mathbf{Y}^l$ and $\mathbf{Y}^h$ are the matrices representing, respectively, the whole LR and HR dictionaries.

Differently than local learning methods, in sparse coding methods there is no explicit nearest neighbor search, but the selection of the dictionary elements participating in the patch combinations is concurrent with the error minimization process. Moreover, this selection can be "adaptive": the number of combining patches (i.e. the number of nonzero elements of the weight vectors α_i, $\|\alpha_i\|_0$) is

not necessarily fixed, as we have with number of neighbors in the NNS, but can vary according to the approximation error.

A big advantage of sparse coding methods is the possibility to learn a much more compact dictionary based on sparse signal representation, thus limiting the size of the external dictionary, a notable problem for example-based SR. To this end, Yang *et al.* propose in [67, 52] a sparse learning approach, where the dictionary matrix $\mathbf{D}$ is seen as a matrix of training signals. This matrix is composed of the initial dictionary patches; the goal is to factor it into the product of a new dictionary matrix $\mathbf{D}_{new}$ and a representation matrix $\mathbf{A}$, under the constraint that $\mathbf{A}$ has a sparse column-wise form. The two disjointed learning problems, for the LR and HR dictionaries can then be expressed as follow:

$$\begin{aligned} \mathbf{D}^l_{new} &= \underset{\{\mathbf{D}^l,\mathbf{A}\}}{\arg\min} \|\mathbf{Y}^l - \mathbf{D}^l\mathbf{A}\|_2^2 + \lambda\|\mathbf{A}\|_1 \\ \mathbf{D}^h_{new} &= \underset{\{\mathbf{D}^h,\mathbf{A}\}}{\arg\min} \|\mathbf{Y}^h - \mathbf{D}^h\mathbf{A}\|_2^2 + \lambda\|\mathbf{A}\|_1 \end{aligned} \quad . \tag{1.18}$$

In [67, 52] the authors propose to perform a joint learning of the two dictionaries, by forcing the LR and HR bases to share the same code. The goal of the joint learning is to enforce the similarity of sparse representations between a LR and HR image patch pair w.r.t. their own dictionaries. The two equations in (1.18) result then in the following unique formulation:

$$\mathbf{D}_{new} = \underset{\{\mathbf{D},\mathbf{A}\}}{\arg\min} \|\mathbf{Y} - \mathbf{D}\mathbf{A}\|_2^2 + \hat{\lambda}\|\mathbf{A}\|_1 \ , \tag{1.19}$$

where $\mathbf{Y} = \left[\frac{1}{\sqrt{M^h}}\mathbf{Y}^h; \frac{1}{\sqrt{M^l}}\mathbf{Y}^l\right]$ is the unified matrix of training signals, and $\mathbf{D} = \left[\frac{1}{\sqrt{M^h}}\mathbf{D}^h; \frac{1}{\sqrt{M^l}}\mathbf{D}^l\right]$ is the whole dictionary, with the dictionaries concatenated (M^l and M^h are the dimensions of, respectively, the LR and HR patch vectors).

To solve the sparse learning problem (1.19), several algorithms in the literature can be employed, e.g. *K-SVD* (K-singular value decomposition) [68]. These algorithms typically present a two-step procedure: the basis matrix $\mathbf{D}$ and the representation matrix $\mathbf{A}$ are alternatively updated to progressively converge towards the desired solution. In the implementation of the algorithm of Yang *et al.* [52], the feature-sign algorithm of [69] is used to perform the sparse coding step. To update the bases of the new dictionary the authors use the Lagrange dual algorithm proposed by the same authors in [69]. Once the new dictionary $\mathbf{D}_{new}$ is trained, $\mathbf{D}^l_{new}$ and $\mathbf{D}^h_{new}$ are retrieved by simply splitting the unique learned matrix, and can be straightly used for SR reconstructions.

Within the work done in order to accomplish this doctorate, we had the chance to also work on dictionary learning problems, not necessarily related to SR applications. In particular, we designed a new algorithm to factor a nonnegative matrix

of training signals with a special sparsity constraint on the factor representing the representation matrix, thus learning a nonnegative dictionary suitable for sparse representations. The algorithm, which takes the name of *K-WEB*, is illustrated in Section 5.3.5. K-WEB is tested for problems related to image approximation and dictionary recovery.

Chapter 2

Single-image SR based on nonnegative neighbor embedding and external dictionary

This chapter presents our main contributions [1] on single-image example-based super-resolution (briefly indicated as "example-based SR"). Basing on the analysis we provided in Section 1.2, there are two discriminating aspects in example-based SR: the type of dictionary employed and the patch reconstruction method used in the patch-based SR procedure. In this chapter, we design algorithms that make use of an *external dictionary* and neighbor embedding as patch reconstruction method.

The algorithms presented are the result of several contributions, according to a progressive improvement of an initial proposal. Starting from the general neighbor embedding SR procedure, which is described in Section 2.1, we propose a novel procedure based on nonnegative neighbor embedding (Section 2.2). A new method to construct a more compact and performing dictionary to be used with the neighbor embedding procedure proposed is then presented in Section 2.3. Finally, a new enhanced interpolation tool and new training schemes for extracting the patches are introduced in Section 2.4, thus leading to the final version of the algorithm.

2.1 Neighbor embedding based SR

Example-based single-image SR aims at finding a HR output image I_H, given a LR input image I_L and a dictionary of training examples, usually in the form of patches, $\mathcal{D}$. The SR procedure consists in single patch reconstructions: the

1. The contributions presented in this chapter appeared in the papers [73, 64, 78, 82].

HR output image is reconstructed "piece by piece", each piece corresponding to a certain patch in the LR input image. In Section 1.2.2.1, we presented *neighbor embedding* (NE) as a possible patch reconstruction technique, within the family of local learning methods. We can then define a whole NE-based SR procedure, which follows the example-based SR paradigm and adopts neighbor embedding to estimate the HR patches.

As mentioned, in the example-based approach we have a dictionary $\mathcal{D} = \left(\mathcal{Y}^l, \mathcal{Y}^h\right)$, formed by several LR/HR patch co-occurrences, where $\mathcal{Y}^l = \{\mathbf{y}_i^l\}_{i=1}^{N_y}$ is a set of LR patch vectors and $\mathcal{Y}^h = \{\mathbf{y}_i^h\}_{i=1}^{N_y}$ is the set related to their HR counterparts. The basic idea of NE is that we can express a LR input patch as the weighted combination of its K K nearest neighbors (K-NN) selected from the dictionary, and then apply the same weighted combination to the corresponding HR patches in the dictionary to reconstruct the HR output patch. All patches, both from the dictionary and the test images, are transformed into *feature vectors*, by concatenating some features computed on the pixels of the patches. Generally, we can have two distinct feature representations for the LR and the HR patches. In Section 2.1.2 we revise this aspect of NE-based super-resolution.

When super-resolving a LR input image I_L, the first step is to divide I_L into patches of the same size of the LR patches in the dictionary, and convert them into feature vectors, so obtaining the set of LR feature vectors $\mathcal{X}^l = \{\mathbf{x}_i^l\}_{i=1}^{N_x}$. The goal of the neighbor embedding algorithm is to produce, from each single LR input patch, a HR output patch, by taking into account K patches in the dictionary. Finally, we obtain a set of HR feature vectors $\mathcal{X}^h = \{\mathbf{x}_i^h\}_{i=1}^{N_x}$, from which we can reconstruct the entire output image. The steps of the NE-based SR approach can be briefly summarized as follows.

1. For each LR patch feature vector $\mathbf{x}_i^l \in \mathcal{X}^l$
   (a) Find its K-NN in $\mathcal{X}_d$ in terms of Euclidean distance:
   $$\mathcal{N}_i = \underset{\mathbf{y}_j^l \in \mathcal{Y}^l}{\arg\min^K} \left\|\mathbf{x}_i^l - \mathbf{y}_j^l\right\|^2 \ . \tag{2.1}$$
   (b) Find a weighted combination that approximates $\mathbf{x}_i^l$ with the selected neighbors, i.e. compute the K weights $\{w_{ij}\}_{j=1}^K$ such that:
   $$\mathbf{x}_i^l \approx \sum_{\mathbf{y}_j^l \in \mathcal{N}_i} w_{ij} \mathbf{y}_j^l \ . \tag{2.2}$$
   (c) Apply the same weights for the reconstruction of the output HR patch feature vector $\mathbf{x}_i^h$ with the corresponding neighbors in $\mathcal{Y}^h$:
   $$\mathbf{x}_i^h = \sum_{\mathbf{x}_j^h \in \mathcal{H}(\mathcal{N}_i)} w_{ij} \mathbf{x}_j^h \ . \tag{2.3}$$

where $\mathcal{H}(\mathcal{N}_i)$ indicates the set of HR feature vectors in the dictionary corresponding to the LR neighborhood $\mathcal{N}_i$.

2. Once all the HR patch feature vectors are generated, reverse them back to pixel-based patches, and combine the obtained patches, by simply averaging the pixel values in the overlapping regions, to form the output image.

In the neighbor embedding SR framework there are two main choices to make: the method applied to perform the neighbor embedding, i.e. compute the common weights of each linear combination, and the feature representations used for the LR and the HR patches. As for the former, in next section we introduce the original neighbor embedding algorithm based on Locally Linear Embedding (LLE), adopted by Chang *et al.* [49]. An overview about the concept of patch features is provided in Section 2.1.2.

2.1.1 LLE-based neighbor embedding

As a first work in NE-based super-resolution, Chang *et al.* proposed in [49] a novel SR approach based on neighbor embedding. The basic idea of their algorithm is inspired by a method for data dimensionality reduction called *Locally Linear Embedding* (LLE) [63, 70]. LLE is a manifold learning method: namely, it relies on the assumption that data approximately lie on a lower-dimensional *manifold* (Figure 2.1); to perform the dimensionality reduction, then, it is sufficient to "discover" this manifold. To do that, LLE implements a simple geometric intuition: provided that there are sufficient data such that the manifold is well-sampled, since the manifold is locally linear, the geometry around a certain point can be well represented by the linear combination of the point itself through its nearest neighbors. The dimensionality reduction is then performed preserving the local relations found, i.e. the weights of each linear combination.

In [49] Chang *et al.* adapted the philosophy of LLE to the SR problem: assuming that patches in the LR and HR images form manifolds with similar local geometries (possibly in two "feature spaces"), then it is possible to express a given HR output patch as the combination of its neighbors laying on the HR manifold, using the same weights of the linear combination that approximates the corresponding LR patch. The algorithm does not completely apply LLE, since there is no dimensionality reduction performance; it takes from it only the weight computation step, specially constructed to best represent a point from its nearest neighbors, by minimizing the Euclidean norm of the approximation error. The "manifold similarity" between LR and HR patches, that was the strong motivation for the authors to use multiple neighbors to reconstruct an output patch, however, is not generally valid, as pointed out in [71]: if some LR patches are neighbors each other, their corresponding HR patches are not necessarily in neighborhood anymore.

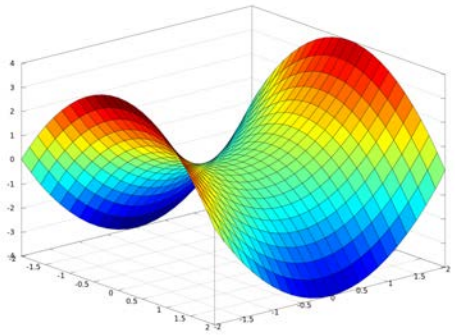


Figure 2.1: Example of manifold - A manifold of dimension K in $\mathcal{R}^N$ is a subset of $\mathcal{R}^N$ which "looks locally" like an Euclidean subspace $\mathcal{R}^K$ of $\mathcal{R}^N$.

As said, the weights of each linear combination (Step 1b of the procedure presented in the previous section), are computed by minimizing the approximation error of the related LR patch, $\varepsilon^i = \|\mathbf{x}_i^l - \sum_{\mathbf{y}_j^l \in \mathcal{N}_i} w_{ij}\mathbf{y}_j^l\|^2 = \|\mathbf{x}_i^l - \mathbf{Y}_i^l\mathbf{w}_i\|^2$. Minimizing the Euclidean norm of this error leads to a least squares (LS) approximation problem. Computing a linear combination of neighbors to approximate a particular data point is equivalent to learning the hyperplane that locally approximates the manifold. This operation needs to be independent of a particular frame of reference; therefore the weights computed need to be invariant to rotations, rescalings and translations of the data point and its neighbors. The first two conditions follow immediately from the formulation of the approximation error. As for the invariance to translations, it can be easily proved that this can be achieved by enforcing a sum-to-one constraint for the weights.

We then have the following constrained least squares (LS) minimization problem (*SUM1-LS*):

$$\mathbf{w}_i = \arg\min_{\mathbf{w}} \|\mathbf{x}_i^l - \mathbf{Y}_i^l\mathbf{w}\|^2 \quad \text{s.t.} \quad \mathbf{1}^T\mathbf{w} = 1 \ . \tag{2.4}$$

A solution to (2.4) can be found through the method of Lagrange multipliers, and it is:

$$\mathbf{w}_i = \frac{\mathbf{G}_i^{-1}\mathbf{1}}{\mathbf{1}^{\mathrm{T}}\mathbf{G}_i^{-1}\mathbf{1}} \ , \tag{2.5}$$

where the matrix $\mathbf{G}_i^{-1} := (\mathbf{x}_i^l\mathbf{1}^{\mathrm{T}} - \mathbf{Y}_i^l)^{\mathrm{T}}(\mathbf{x}_i^l\mathbf{1}^{\mathrm{T}} - \mathbf{Y}_i^l)$. The proof of 2.5 is given in Appendix A.

The "SUM1-LS" weights are used in the original NE-based super-resolution algorithm in [49], as well as in several other example-based SR algorithms (e.g. [50, 51]). In Section 2.2 we propose an alternative criterion to compute the weights

of the neighbor embedding and study the issue of the feature representation. This will lead us to the formulation of a new NE-based SR algorithm.

2.1.2 Feature representation

Color images are typically represented by the RGB color model. An alternative representation is to use a luminance-chrominance model, e.g. the YIQ[2] 3-matrix representation used in the NTSC color TV system, where the luminance component Y represents the brightness of the image and the other two components I and Q, called chrominance, convey the color information of the picture. For SR purposes the latter representation is the most convenient. In fact, since humans are more sensitive to changes in the brightness of the image rather than changes in color, only the Y component is super-resolved, so reducing the complexity of the SR algorithm. The chrominance matrices are simply upsized by interpolation.

As discussed before, the entire neighbor embedding SR procedure is carried out in a feature space: LR and HR patches are represented by vectors of features, namely some transformations of the luminance values of the pixels of the patch (see Figure 2.2 for a schematic representation).

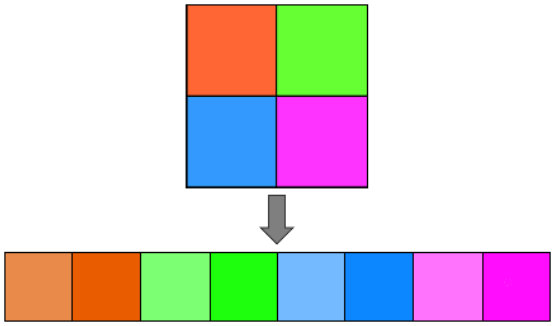


Figure 2.2: Feature representation of a patch - One ore more features are computed for each pixel of the patch; all features are then concatenated into a vector form.

The role of the features is double:

- to catch the most relevant part of the LR information in order to have an as much as possible good "predictor" for the HR patch reconstruction;
- to possibly enforce the similarity between the LR and HR patch manifolds.

In order to pursue the first purpose, several algorithms in the literature, even not based on neighbor embedding, propose to pre-process the LR images (the

2. The luminance-chrominance model is usually denoted as YUV, where YUV is the representation for analog TV with slightly different weights than YIQ.

LR training images and the LR input image) before extracting the LR patches, so obtaining transformed luminance values. E.g. in [48, 53] the LR patches are extracted from a band-passed version of the related LR image (previously interpolated). In [72, 50] image "primitive" patches, obtained by convolving the LR image with a bank of Gaussian derivative filters, are taken as LR patches. In the original neighbor embedding algorithm [50] and in [52], instead, simple features derived from the first and second-order gradient are taken into account. In this case, first and second-order derivatives in both directions are computed for each pixel of the patch (4 values per pixel). The gradient values are computed by simply applying these four 1-D filters to the luminance matrix:

$$\begin{array}{ll} g_1 = [-1, 0, 1], & g_2 = g_1^T \\ g_3 = [1, 0, -2, 0, 1], & g_4 = g_3^T \end{array} . \tag{2.6}$$

In all cases the idea is that the middle and high-frequency content of the LR patches is the most helpful information to learn the LR-HR patch correspondences. As for the representation of the HR patches, ideally, in accordance with the principle of manifold similarity, we would like to keep the same feature space as for the LR patches. However, the feature representation chosen is constrained to the fact that at the end of the SR algorithm we have to reverse the features back to pixel-based patches. This reversion step is not feasible in the case of gradient feature vectors (we have more "unconstrained" equations than unknowns) or even more complicated features. Then, the common solution adopted in the literature is to straight use the luminance values of the HR training images, possibly after doing an operation of contrast normalization or mean removal.

2.2 Example-based SR via nonnegative neighbor embedding

In this section[3] we present a new neighbor embedding (NE) based algorithm, which reconstructs the HR output image by means of nonnegative combinations of patches taken from an external dictionary. Starting from the original NE-based algorithm of Chang *et al.* [49], which makes use of what we called the "SUM1-LS weights", and testing it with different feature representations, in Section 2.2.1 we deduce the necessity of introducing a different weight computation method. In Section 2.2.2, then, we propose new weights for the SR problem based on non-negative neighbor embedding. The deriving algorithm is fully illustrated in Section 2.2.3, and its results are finally presented in Section 2.2.4.

3. The work described in this section has been presented in two papers of ours, [73, 64].

2.2.1 Analysis of the features and criticality

In order to evaluate the performance of the NE-based SR approach in presence of different feature representations, we implemented the original LLE-based neighbor embedding algorithm of Chang *et al.* [49] and tested it with the following three feature configurations for the LR patches.

F 1 *First order gradient*: first-order derivatives in both directions are computed for each pixel of the patch (2 values per pixel), by applying to the luminance matrix $g_1 = [-1, 0, 1]$ and $g_2 = g_1^T$ as 1-D filters. Let N be the number of pixels in a patch and d the dimension of the LR feature vectors, then $d = 2N$.

F 2 *Centered luminance values*: the features are obtained by taking the luminance values and subtracting the mean value. This corresponds to remove the DC component of the patch; we can therefore see also this method as providing a low-cut filtered version of the LR patch. In this case, $d = N$.

F 3 *F1+F2*: concatenation of F1 and F2 (thus, $d = 3N$) as considered in [51].

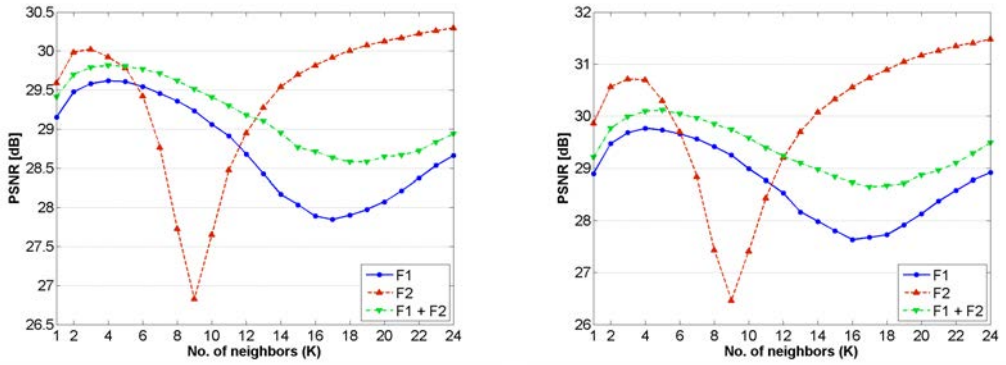
As for the HR patches, as explained, we are obliged to use luminance-based features, in order to make the reconstruction of the patch pixel values possible. Given this, we choose to use centered features (F2) in any configuration: in this case, the HR features retrieved are revertible, by simply adding the mean value of the corresponding LR patches in the input image. Table 2.1 summarizes the features configurations (LR and HR patches) used in the tests.

LR features	**HR features**
First order gradient (F1)	Centered luma values (F2)
Centered luma values (F2)	Centered luma values (F2)
F1+F2	Centered luma values (F2)

Table 2.1: Feature representations used for LR and HR patches - For the LR patches, we tested three different kinds of features; for the HR patches the features used are unique in order to make the re-conversion to luminance-based values possible

Among the 3 possible options for the LR features (F1, F2, F1+F2), we expect centered luminance values (F2) to be the most intuitive solution, as the representation for LR and HR patches would be unique. Moreover, this is the "cheapest" solution in terms of computational time, as the feature vectors contain less entries than for the other feature representations. In [51] a performance analysis of the feature representations is given, including gradient features and centered luminance values ("norm luminance"): the authors conclude that the best solution for the LR features is a weighted concatenation of norm luminance and first-order

gradient values, while purely centered luminance values representing the fourth or fifth choice (see Figure 3 in [51]). However, this analysis is not performed by considering variations of K (number of neighbors). Figure 2.3 evaluates for two test images ("head" and "bird") the performance of the NE-based SR algorithm (*PSNR* of the super-resolved image) versus K, with the chosen LR features and the standard *SUM1-LS* method used to compute the weights.



head bird

Figure 2.3: Comparison between LR feature representations, *SUM1-LS* used as NE method - For both test images considered, the curves present an "up-down-up" behavior.

Figure 2.3 shows that the performance of the algorithm is highly dependent on the number of neighbors K. For all the feature representations, we observe that the curves present an "up-down-up" behavior; the fall in the case of centered features appears even dramatic. We explain this behavior with the "blindness" of the NE approach: the weights of the patch combinations are computed on the LR feature vectors and then blindly applied to the corresponding HR vectors. As we observe, a problem arises at a certain critical point that depends on the dimension of the LR vectors d. For F2, we can make the following observations.

Observation 2.2.1 *Let d be the dimension of the LR vectors, K the number of neighbors, and r_i the rank of the ith neighborhood matrix $\mathbf{Y}_i^l$. Then, the neighbors (centered) vectors lie on a $d-1$-dimensional hyperplane, and the rank of the neighborhood matrix is upper-bounded as follows: $r_i \leq min(d-1, K)$. For the dictionary built as in Section 2.2.3, we observe that the upper bound is tight. More precisely, $r_i = min(d-1, K)$ with high probability.*

Observation 2.2.2 *For $K = d$ the SUM1-LS problem is assimilable to the solution of a square linear system, as we have d equations (the $d-1$ linearly independent equations of $\mathbf{x}_i^l \approx \mathbf{Y}_i^l \mathbf{w}_i$ plus the one given by the equality constraint) in $K = d$ unknowns. Here, experimentally we have a drop in the performance.*

Intuitively, we explain this criticality with the fact that the LS solution is "too fitted" on the LR data and thus generates undesired bad HR reconstructions. In Figure 2.3 this point is, in case of F2, at $K = d = 9$, as we use 3×3 LR patches.

Nevertheless, we observe that outside the fall region, centered luminance features outperform the other feature representations, thus showing the expected potential. Therefore, we decide to use F2 as unique features for LR and HR patches. In the following section we propose a new criterion for the weight computation, in order to avoid the irregular performance with K and fully exploit the common features.

2.2.2 A nonnegative embedding

In the previous section we pointed out that, for any feature chosen, we have a fall of the performance for a certain critical point, by using *SUM1-LS* as a NE method.

We believe that this problem can be avoided by replacing the sum-to-one equality constraint by a more "relaxed" inequality constraint. Therefore, we propose another method for computing the neighbor embedding, derived from a LS problem equivalent to (2.4), but with a non-negativity inequality constraint, according to "the intuitive notion of combining parts to form a whole" [74] (only additive combinations of patches are allowed):

$$\mathbf{w}_i = \arg\min_{\mathbf{w}} \|\mathbf{x}_i^l - \mathbf{Y}_i^l \mathbf{w}\|^2 \quad \text{s.t.} \quad \mathbf{w} \geq 0 \; . \tag{2.7}$$

The problem in (2.7) is known in the literature as non-negative least squares (NNLS); an iterative solution is provided in [75, Ch. 23, p. 161]. The formula experimentally turns out to converge after an average of 2 iterations.

As an alternative and possibly faster solver of (2.7), in [73] we also proposed to use Semi-nonnegative Matrix Factorization (SNMF) [76]. In fact, SNMF is a method to perform a factorization of a matrix, where, differently from "full" Nonnegative Matrix Factorization (NMF) [74], only one factor is constrained to have positive values. The aimed factorization is in the form:

$$X \approx FG^T \; , \tag{2.8}$$

where we restrict G to be nonnegative, while placing no restriction on the signs of F. In the neighbor embedding case, we can reformulate (2.8) as $\mathbf{x}_i^l \approx \mathbf{Y}_i^l \mathbf{w}_i$, where the matrix to be approximated is a patch of the LR input image, the unconstrained factor is the matrix formed by its neighbors ($\mathbf{Y}_i^l$), and the nonnegative factor is the weight vector we want to compute.

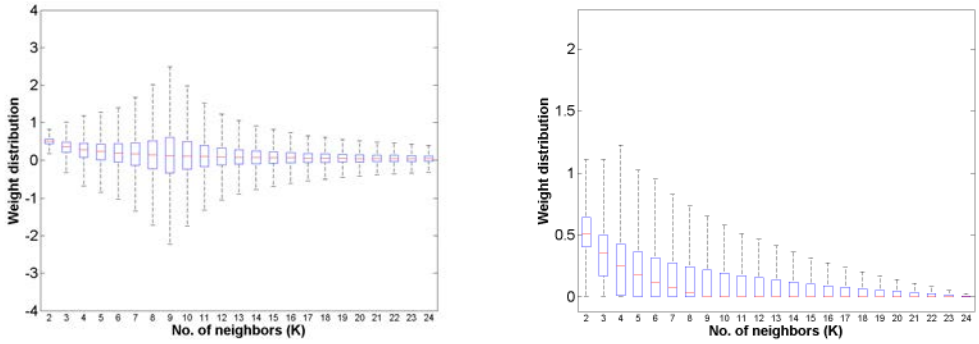
In [76] an iterative solution to (2.8), that is proved to converge to a local minimum of the Euclidean distance $\|X - FG^T\|^2$, is found. The iterative solution

consists of two multiplicative update rules, both for F and G elements. In our case, however, the matrix F, formed by the actual LR patches in the dictionary, is *fixed* and no update rule is needed for it. We then implement only the rule regarding the weight vector, which, with our notation, can be rewritten in the following way:

$$w_{ij} \leftarrow w_{ij} \sqrt{\frac{\left(\mathbf{x}_i^{l\,T}\mathbf{Y}_i^l\right)_j^+ + \left[\mathbf{w}_i{}^T\left(\mathbf{Y}_i^{l\,T}\mathbf{Y}_i^l\right)^-\right]_j}{\left(\mathbf{x}_i^{l\,T}\mathbf{Y}_i^l\right)_j^- + \left[\mathbf{w}_i{}^T\left(\mathbf{Y}_i^{l\,T}\mathbf{Y}_i^l\right)^+\right]_j}}\,, \tag{2.9}$$

where the positive and negative parts of a matrix A are defined respectively as $A_{ik}^+ = (|A_{ik}| + A_{ik})/2$ and $A_{ik}^- = (|A_{ik}| - A_{ik})/2$. The convergence of the formula, i.e. the non-increase of the approximation error $\|\mathbf{x}_i^l - \mathbf{Y}_i^l\mathbf{w}_i\|^2$ while updating only the weight vector, can be proved in the same way as in [76]. Thus, (2.9) can be used as an iterative formula to obtain nonnegative weights for each patch combination.

In Figure 2.4 the weight distributions of the two NE criteria are compared, using the box plot representation. Each box is delimited by the 25th (Q1) and 75th percentile (Q3). Values that exceed the boxes by $2\times(Q3-Q1)$ are considered outliers and not drawn. Interestingly, the values for *SUM1-LS* weights get larger in the proximity of the critical points; instead, the values of the *NNLS* weights regularly decrease with K.



SUM1-LS NNLS

Figure 2.4: Distribution of the NE weights for each value of K - Centered luminance values (F2) are used as LR features.

To further evaluate the goodness of the new nonnegative NE criterion compared to *SUM1-LS*, we used the distance (in terms of MSE) between the LR weights, resulted by the approximation of a LR test patch with its LR neighbors, and the HR weights, obtained by approximating the corresponding HR test patch

with the HR vectors related to the LR neighbors found. This is an index of how the LR weights, i.e. what we can actually compute, reflect the ideal HR weights. We averaged the error over 1000 randomly selected test patches and tested two different dictionary of training patches (Figure 2.5).

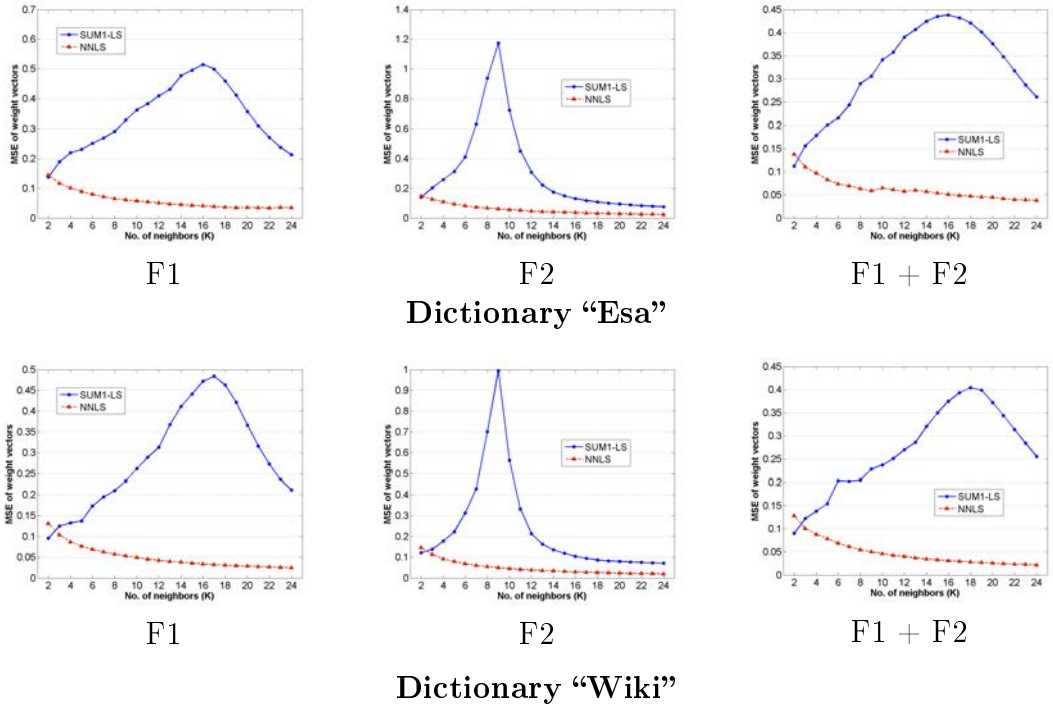
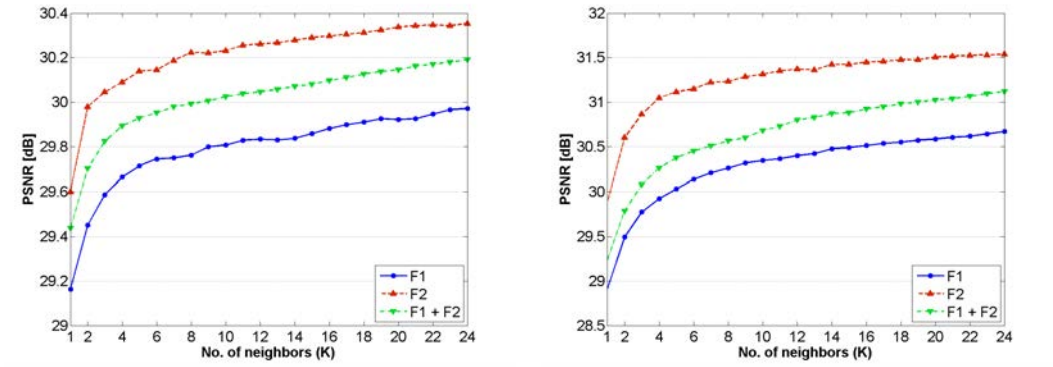


Figure 2.5: Distance between LR and HR weights for *SUM1-LS* and *NNLS* - The distance, averaged over 1000 patch reconstructions, is measured in terms of MSE of the two weight vectors; two different "sources" of neighbors, i.e. two dictionaries ("Esa" and "Wiki"), are taken into account.

Figure 2.5 shows that the new nonnegative embedding method is much more suitable for computing the weights of a common embedding, as the MSE error between the weight vectors is, in any case, below the corresponding one for *SUM1-lS*. The results are confirmed in Figure 2.6, where a comparison between different features (in terms of ouput *PSNR*), while using *NNLS*, is presented.

Compared to Figure 2.3, Figure 2.6 presents a much more stable performance, as K varies: the *PSNR* curve is monotonically increasing. Moreover, the *PSNR* values are generally higher and F2 are clearly the winning features.



head bird

Figure 2.6: Comparison between LR feature representations, *NNLS* used as NE method.

2.2.3 NonNegative Neighbor Embedding (NoNNE) SR algorithm

From the analysis undertaken in the previous sections, we can derive a new NE-based SR algorithm: centered luminance values (F2) are taken as features for both LR and HR patches, and the new NE method with nonnegative weights is chosen to compute the patch combinations. Hereafter, we refer to this algorithm as *NoNNE* (Nonnegative Neighbor Embedding).

An important issue for example-based SR algorithms is represented by the choice of the dictionary. To pursue the aim of realizing a low-complexity algorithm, we decide to adopt a one-step upscaling procedure, whereas other algorithms (e.g. [58, 56]) achieve the desired magnification by several upscalings with smaller scale factors (the SR procedure is thus iterated several times). For the dictionary, we have then two possibilities:

1. Build an external dictionary from a set of training images (from the original HR images, generate the LR versions, and extract HR and LR patches, respectively), and
2. Learn the patch correspondences in a pyramid of recursively scaled images, starting from the LR input image, in the way of [58] (see also Figure 1.4).

In Table 2.2 results from the pyramid internal dictionary and two external dictionaries are reported; *NNLS* with $K = 12$ is used as the NE method. For both the external dictionaries, a small number of natural images is taken. As we can see from the table, the external dictionary performs significantly better, since the number of pairs of patches we can derive from the internal pyramid is insufficient. Note, in fact, that in a pyramid of image layers like in Figure 1.4, if the SR procedure must consist of only one step, only the pyramid layers whose distance is

equal to the total magnification factor can be coupled and used as training images. Thus, we cannot use many image pairs in the pyramid to build dictionary. On the other hand, the size of the external dictionary can be tuned to any value, by conveniently choosing the training images. In the following experiments, we build a dictionary of a size s.t. the upscaling of a 70×70 image by a factor of 4 takes about 5 seconds.

		Internal DB		**Ext DB "Esa"**		**Ext DB "Wiki"**	
Image	**Scale**	*PSNR*	DB size	*PSNR*	DB size	*PSNR*	DB size
Head	4	28.62	525	30.22	56514	30.24	218466
Baby	4	27.95	2179	30.62	56514	30.32	218466
Eyetest	3	16.10	3827	18.38	100966	18.10	389232
Bird	3	27.72	2256	31.37	100966	31.42	389232
Woman	2	27.29	15044	30.91	229096	30.44	880440

Table 2.2: Final *PSNR* and DB size for different dictionaries - The DB size is intended as the number of pairs of patches ("DB" stands for database, i.e. dictionary).

There is one important parameter left to set in the algorithm: the size of the LR patches taken from the LR input image. From our experiments we found that the optimal size is 3×3 with a 2-pixel overlap. As we perform a one-step upscaling, the size and the overlap degree of the HR patches come as consequences of the magnification factor: e.g. with a factor of 4 we will have 12×12 HR patches with a 8-pixel overlap. The reason for choosing largely overlapping HR patches is due to the fact that at the end of the NE algorithm (see Step 2 of the general procedure in Section 2.1) we combine the patches together by simply averaging the related pixel values in the overlapping regions. By having big overlaps, we can in fact implicitly impose smooth properties on the output image. Moreover, as an additional tool used by several SR algorithms (e.g. [58]), we implement a final operation to assure the super-resolved image to be consistent with the LR input image: an *iterative back projection* (IBP). Note that the method in [58] uses IBP, that stops on average after 3 iterations, at each step; we use it only once instead. This method consists in "back-projecting" the obtained HR image $\hat{I}_H$ to a corresponding LR image $\hat{I}_L$, by blurring and downsizing it, i.e. in reversing the SR process. This LR image is supposed to be similar to the LR input image I_L: therefore, we can compute an error as the difference between the two images, and use this error to correct the output. *IBP* is a particularly useful tool in case of mismatch between the blur kernel used to generate the dictionary and the unknown blurring effect the input image has been affected with. We provide a wider insight of IBP in Section 2.4, where this tool is more extensively exploited.

To sum up, starting from the general NE-based SR procedure described in

Section 2.1, we derived a new example-based single-image SR algorithm, which we called *NoNNE*. This is the result of two fundamental changes, concerning the feature system used (centered luminance values both for LR and HR patches) and the weight computation method (the new nonnegative neighbor embedding). IBP is also performed as a last operation in the algorithm. The complete procedure, reported in listing 1, is very simple but stable (the *PSNR* curve is stricly increasing with K and no fall in the performance is observed). In fact, the only internal parameter to set is the number of neighbor K. However, as shown in Figure 2.6, the *NNLS* method is not much sensitive to it.

Listing 1 NE-based SR with nonnegative weights and centered features

procedure NoNNE(I_L,$\mathcal{Y}^l$,$\mathcal{Y}^h$,K)
  Divide I_L into 3×3 patches (with a 2-pixel overlap)
  $\mathcal{X}^l = \{\mathbf{x}_i^l\}_{i=1}^{N_x}$ ▷ **Compute centered features**
  $\{\overline{x}_i^l\}_{i=1}^{N_x}$ ▷ **Store mean values of the LR patches**

  for $i \leftarrow 1, N_x$ **do**
    $\mathcal{N}_i = \underset{\mathbf{y}_j^l \in \mathcal{Y}^l}{\arg\min^K} \left\|\mathbf{x}_i^l - \mathbf{y}_j^l\right\|^2$ ▷ **Find K-NN in $\mathcal{Y}^l$**
    $\mathbf{w}_i = \underset{\mathbf{w}}{\arg\min} \left\|\mathbf{x}_i^l - \mathbf{Y}_i^l\mathbf{w}\right\|^2 \quad \text{s.t.} \quad \mathbf{w} \geq 0$ ▷ **Solve *NNLS***
    $\mathbf{x}_i^h = \mathbf{w}_i\mathbf{Y}_i^h$ ▷ **Compute the HR feature vector**
  end for

  $\mathbf{x}_i^h \leftarrow \mathbf{x}_i^h + \overline{x}_i^l\mathbf{1} \quad i = 1, \ldots, N_x$ ▷ **Re-add the stored means**
  Rearrange the HR vectors into square patches
  Combine together the HR patches to form the HR output image $\hat{I}_H$
  Iterative back projection to refine $\hat{I}_H$
end procedure

2.2.4 Results with NoNNE

In this section some visual results and comparisons between the *NoNNE* algorithm and other methods in the state of the art are presented. In particular, our algorithm is compared to the original LLE-based NE algorithm of Chang *et al.* [49], to the pyramid-based algorithm of Glasner *et al.* [58] (the results are obtained by using a third party implementation), which is currently considered among the most advanced single-image SR algorithms, and to the Kernel Ridge Regression (KRR) method of Tang *et al.* [65]. The algorithms taken into account for the comparison and their characteristics are summarized in Table 2.3.

Name	Procedure	LR features	Patch reconstruction method
Chang *et al.*	single-step	gradient (1st-2nd)	NE with SUM1-LS weights
Glasner *et al.*	multi-pass	luminance	NE with exponential weights
Tang *et al.*	single-step	gradient (1st-2nd)	Kernel ridge regression
Our algorithm	single-step	centered luminance	NE with NNLS weights

Table 2.3: Summary of the methods compared to our NoNNE algorithm - For each method the main characteristics (kind of procedure, feature representation, and patch reconstruction technique used) are listed.

Our *NoNNE* algorithm is implemented using *NNLS* as the weight computation method and centered luminance values as features, as described in Section 2.2.3, and taking $K = 15$ neighbors. The original LLE-based algorithm is implemented with the *SUM1-LS* method and gradient features, as in [49]; the number of neighbors chosen is $K = 5$. The results of all comparisons (output *PSNR* and running time in seconds as an indication of the complexity of the algorithm), for 5 different images and 3 magnification factors (2, 3, 4), are reported in Table 2.4. In the experiments, in order to have a performance metric, we start from a ground-truth image and generate from it the LR input image according to the image generation process of 1.8. The variance of the Gaussian blur is set, from time to time, equal to the square root of the scale factor (e.g., for a scale factor of 3, we have $\sigma^2 = \sqrt{3}$).

Figures 2.7, 2.8, and 2.9, show the comparative visual results, respectively, for the "baby" image (magnified by a factor of 4), the "bird" image (magnified by a factor of 3), and the "woman" image (magnified by a factor of 2).

When comparing our algorithm to Chang *et al.* [49] the visual improvements are evident: our algorithm is able to super-resolve finer details, whereas the results of [49] are often affected by ringing artifacts. The impression is confirmed by reading the *PSNR* values. With respect to the local learning method of Tang *et al.* [65], too, the PSNR achieved by our algorithm are always higher. As for the running time, this turns to be in favor of our algorithm in both comparisons, thanks to the low-complexity choice of the features (F2 instead of gradient features). The comparison with the method of Glasner *et al.* [58] is also satisfying: [58] gives generally higher values of *PSNR*, although our method is better per-

		Our algorithm		Chang et al.		Glasner et al.		Tang et al.	
Image	**Scale**	***PSNR***	**Time**	***PSNR***	**Time**	***PSNR***	**Time**	***PSNR***	**Time**
baby	2	34.64	58	33.42	339	34.66	4083	33.72	425
bird	2	34.69	18	32.94	110	34.42	406	33.31	132
butterfly	2	27.54	17	25.90	77	26.83	265	26.05	82
head	2	32.88	18	32.34	145	32.68	367	32.43	151
woman	2	30.91	15	29.43	114	30.61	410	29.64	128
baby	3	32.44	27	31.00	116	32.94	2188	31.47	111
bird	3	31.37	9	29.71	47	32.16	281	30.07	42
butterfly	3	24.31	9	22.58	34	25.66	232	22.72	25
head	3	31.46	12	30.82	68	31.69	370	30.95	54
woman	3	27.98	12	26.45	37	28.79	248	26.66	37
baby	4	30.62	22	29.27	86	31.41	4381	29.70	81
bird	4	28.99	6	27.37	21	30.07	475	27.84	22
butterfly	4	22.05	7	20.50	18	23.94	315	20.61	13
head	4	30.26	6	29.57	26	30.86	379	29.83	28
woman	4	25.66	5	24.25	17	26.79	401	24.46	20

Table 2.4: Results (*PSNR* and running time in sec.) for different images.

forming for a magnification factor equal to 2. Nevertheless, the visual results are fairly comparable. What represents an issue for [58] is the complexity, as the algorithm involves several steps and the dictionary is iteratively updated by taking patches from the image "pyramid": although the values provided only serve to give an idea of the complexity, the algorithm clearly requires much more time than ours and the running time grows exponentially with the size of the input image.

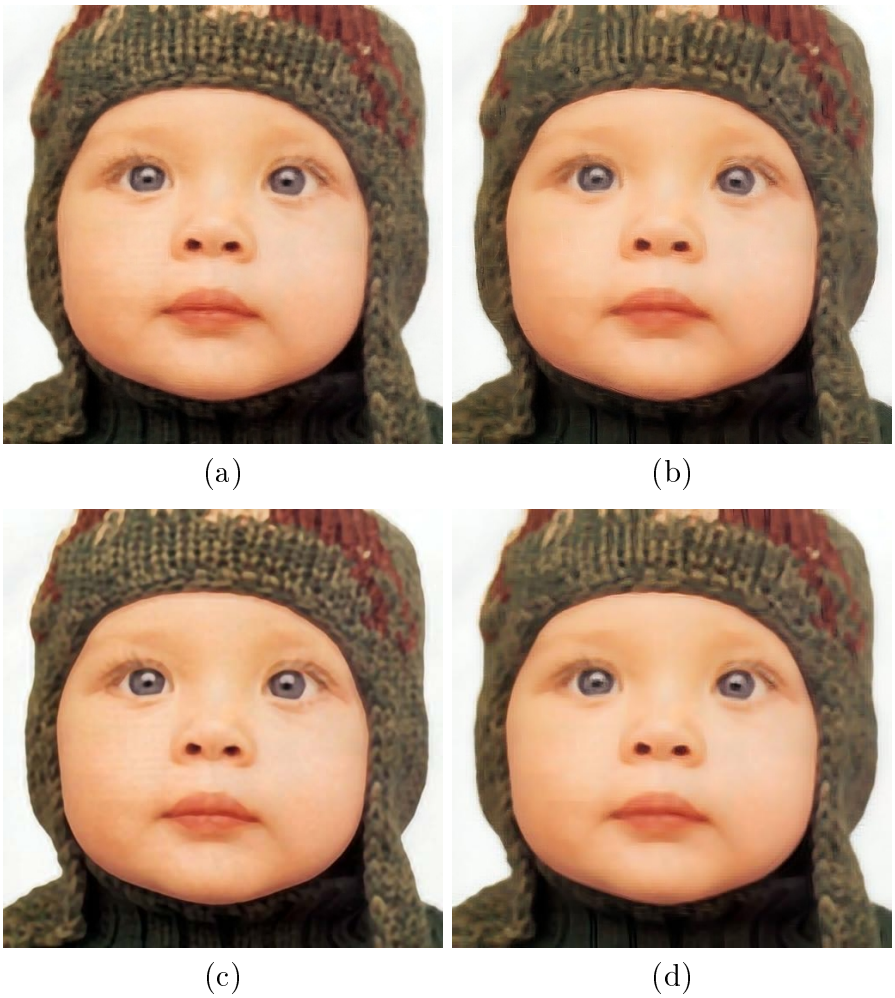
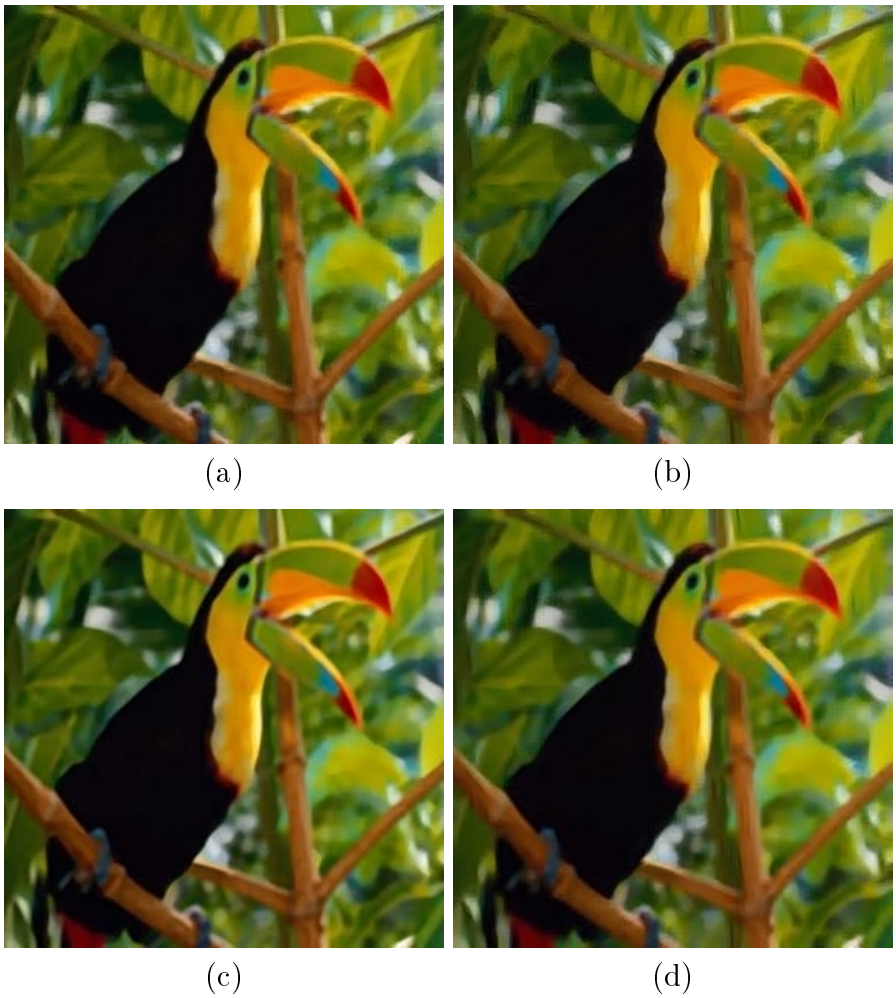


Figure 2.7: Results with the "baby" image (×4) for our *NoNNE* algorithm and other methods - (a) Our algorithm - (b) Chang *et al.* [49] - (c) Glasner *et al.* [58] (7 steps) - (d) Tang *et al.* [65].



(a) (b)

(c) (d)

Figure 2.8: Results with the "bird" image (×3) for our *NoNNE* algorithm and other methods - (a) Our algorithm - (b) Chang *et al.* [49] - (c) Glasner *et al.* [58] (5 steps) - (d) Tang *et al.* [65].

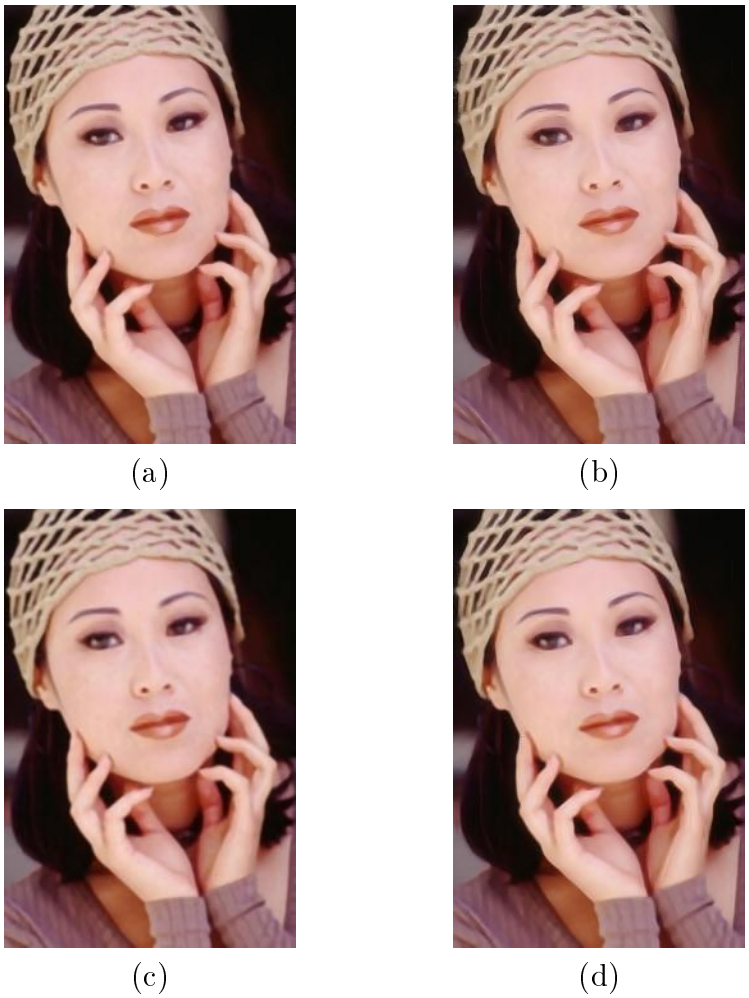


Figure 2.9: Results with the "woman" image (×2) for our *NoNNE* algorithm and other methods - (a) Our algorithm - (b) Chang *et al.* [49] - (c) Glasner *et al.* [58] (4 steps) - (d) Tang *et al.* [65].

2.3 Building a compact and coherent dictionary

This section presents a new method to construct a compact and efficient dictionary for example-based super-resolution (SR) algorithms, i.e. also suitable to the *NoNNE* algorithm we described in Section 2.2. Example-based SR relies on a dictionary of correspondences of low-resolution (LR) and high-resolution (HR) patches. Having a fixed, pre-built, dictionary allows to speed up the SR process, as it does not require any online operation. However a unique dictionary could not be suitable for any input image. To evaluate this problem, we tested the *NoNNE* algorithm with different 5 input images and equally-sized dictionaries (e.g. Figure 2.10 show some of the training images used).

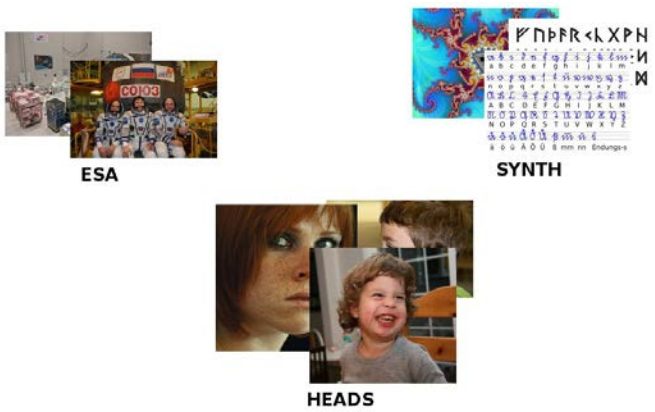


Figure 2.10: Training images used - Three of the dictionaries tested and the related HR images used for the training.

Table 2.5 reports the performance results of the algorithm in terms of *PSNR* of the super-resolved images. As we can see, the results are quite different depending on the dictionary chosen, and the best outcomes for each input image are reached not always with the same one.

In Section 2.3.1 we then propose a new strategy to build a dictionary, by detailing all the steps necessary to construct the final dictionary. The proposed strategy aims at overcoming the problem of the general non-adaptability of external dictionaries, by building a "general-purpose" dictionary. This strategy also takes into account the fact that LR and HR patches often are not coherent, i.e. local LR neighborhoods are not preserved in the HR space. Improving the coherence is taken a "guiding light" in order to perform an intelligent selection of patch pairs. Our designed dictionary construction method takes as input a large dictionary and gives as an output a dictionary with a "sustainable" size, yet presenting comparable or even better performance. It firstly consists of a partitioning process, done according to a joint k-means procedure, which enforces the coherence

		Dictionaries				
Image	**Scale**	**ESA**	**Flower**	**Heads**	**Synth**	**Wiki**
Bird	4	29.51	**29.87**	29.76	28.81	29.12
Butterfly	4	**22.45**	22.00	21.70	21.57	21.98
Eyetest	4	17.52	17.33	17.20	**18.72**	17.13
Head	4	30.55	**30.85**	30.69	30.35	30.42
Newspaper	4	21.93	21.92	**22.01**	21.88	21.68

Table 2.5: *PSNR* values of the *NoNNE* algorithm, tested with 5 different dictionaries - For each input image we have generally a different best-performing dictionary.

between LR and HR patches by discarding those pairs for which we do not find a common cluster. Secondly, the clustered dictionary is used to extract some salient patches that will form the output set. The clustering step is partially similar to the method proposed in [77], where, however, the k-means clustering is done by referring only to the LR patches and is not a joint procedure.

An analysis on the algorithm proposed is provided in Section 2.3.2; Section 2.3.3 shows instead some visual SR results of the dictionary construction procedure applied to the *NoNNE* algorithm, by enlightening the improvements achieved on the latter. The theory and the results presented in this section have been described in a conference paper. [78].

2.3.1 Procedure proposed

An issue with NN-based SR methods, pointed out e.g. in [79], is that selected LR neighborhoods are not preserved when passing to the HR domain, i.e. the HR candidates we actually use to generate the HR output patch are not assured to stay neighbors each others. We call it a lack of "coherence" between the LR and HR patches. This problem is present in any method that requires a NN search, and it is particularly crucial for neighbor embedding based SR methods, where the computed weights are meant to capture the local geometry of the patch neighborhoods.

In Section 2.3.1.1 we propose a new strategy to limit this problem, based on a *joint k-means clustering* of the initial dictionary. In Section 2.3.1.2 we address the problem of designing a compact general-purpose dictionary, thus coming up with a new efficient way of dictionary learning.

2.3.1.1 Joint k-means clustering

In order to have a neighborhood preservation, and thus provide a better coherence between LR and HR patches, we propose to jointly cluster the LR and HR patches of the dictionary, with a procedure similar to the classical k-means approach. We call this algorithm joint k-means clustering (JKC).[4]

The input of the algorithm is a dictionary $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$, formed by N LR/HR patch co-occurrences, where $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ is a set of LR patch vectors and $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^{N}$ is the set related to their HR counterparts. Let k be the chosen number of clusters, we can then define a set of cluster centers $\{\mathbf{c}_j\}_{j=1}^{k}$, where each center $\mathbf{c}_j$ includes a LR and a HR part, respectively $\mathbf{c}_j^x$ and $\mathbf{c}_j^y$. A joint patch vector $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{y}_i)$ is considered to be part of a certain cluster if both $\mathbf{x}_i$ and $\mathbf{y}_i$ share the same center. This principle is explained graphically in Figure 2.11: the motivation of it is to enforce the coherence property, as a pair of patches is checked to actually belong to corresponding LR and HR cluster, basically "larger neighborhoods".

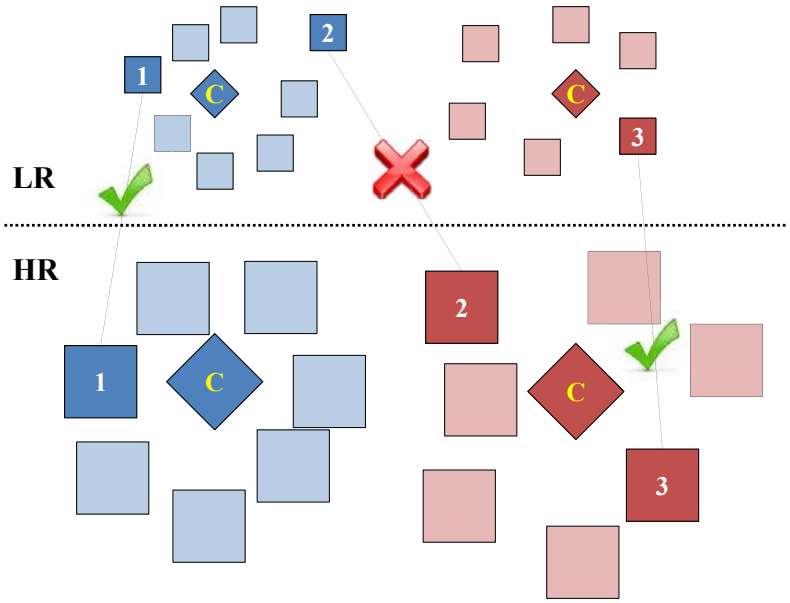


Figure 2.11: Joint k-means clustering (JKC) procedure - The LR and HR patches are partitioned separately, each one into two clusters (there is, though, a one-to-one link between LR and HR clusters. Each pair of patches is then checked to have been assigned to corresponding clusters (as it happens for the pairs 1 and 3). Pair 2 represents instead a mismatch (the LR patch is assigned to the blue cluster, whereas the HR patch is assigned to the red one).

4. In the text "small k" (k) indicates the number of partitions of the clustering process, whereas "capital k" (K) indicates the number of nearest neighbors taken at the patch reconstruction stage of the proper SR algorithm.

To implement this idea we adopt a modified version of the traditional Lloyd's algorithm [80], with the two alternating steps (cluster assignment and recentering), which can be summarized as follows:

1. Arbitrarily initialize the k centers.
2. *(Cluster assignment)* For each $i \in \{1, \dots N\}$, $L(i) = j\prime$ if **both** $\mathbf{c}^x_{j\prime}$ and $\mathbf{c}^y_{j\prime}$ are the closest centers to, respectively, $\mathbf{x}_i$ and $\mathbf{y}_i$; otherwise $L(i) = 0$.
3. *(Cluster re-centering)* For each $j \in \{1, \dots k\}$, we define the related cluster $\mathcal{C}_j = \{\mathbf{z}_i \text{ s.t } L(i) = j\}$ and re-compute the joint center: $\mathbf{c}^x_j = \frac{1}{|\mathcal{C}_j|} \sum_{\mathbf{x}_i \in \mathcal{C}_j} \mathbf{x}_i$ and $\mathbf{c}^y_j = \frac{1}{|\mathcal{C}_j|} \sum_{\mathbf{y}_i \in \mathcal{C}_j} \mathbf{y}_i$.
4. Repeat steps 2 and 3 until L no longer changes.

In the procedure described, L is a vector of labels, which contains, element by element, the index of the assigned cluster. We set $L = 0$ for those vectors that do not find any placement, i.e. do not belong to the same neighborhood (cluster) of LR and HR patches. These vectors are temporarily discarded, i.e. they are put in a "trash cluster", but they are taken into account again at the next iteration when new centers are computed. At the end of the iterative process, however, the pairs belonging to the trash cluster are removed from the dictionary, thus leading to a first pruning of the dictionary.

2.3.1.2 Selection of prototypes as a summary of the dictionary

Table 2.5, as it shows that each dictionary performs more or less well depending on the input image, suggests that we should build the dictionary starting from many images, in order to capture as much variability in the image contents as possible. However, this leads to an increase of the dictionary size, and so of the running time of the SR algorithm. Thus, a strategy to subsequently reduce the dictionary is needed.

We propose to use *JKC* as a starting point to design such a strategy. The jointly clustering procedure, as said, is performed in order to impose the LR and HR patches to share the same neighborhoods, i.e. to respect a sort of "coherency" property. We believe that, if we sample the clustered dictionary by taking those pairs of patches that better respect this property, we would be less effected by the pruning effect, i.e. the decrease of performance due to the fact that we would have a smaller dictionary.

Therefore, to reduce the size of the dictionary, we first get rid of the "trash cluster", i.e. those pairs of patches that did not find a placement after the clustering procedure. For these pairs, the related LR and HR patches are not supposed to be in corresponding LR and HR neighborhoods. We assume that, by eliminating these "bad pairs", we do not loose too much in terms of final results. Secondly, to further decrease the dictionary size, we propose to intelligently sample each

cluster in order to capture as much variety contained in it as possible. We decide then to adopt the strategy of the *K-means++* algorithm [81], where, while randomly choosing the initial centers, the further a point is from the already selected centers, the higher its probability to be selected as a new center. The spirit is to promote diversity in the sampling process. M "prototypes" per cluster are then selected according to the following steps:

1. Select the first prototype uniformly at random among the cluster elements,
2. Compute, for each remaining element $\mathbf{x}$, the distance $D(\mathbf{x})$, i.e. the distance between $\mathbf{x}$ and its nearest prototype,
3. Select one element at random as a new prototype, using a weighted probability distribution where a point $\mathbf{x}$ is chosen with probability proportional to $D(\mathbf{x})^2$, and
4. Repeat steps 2 and 3 until all the M prototypes have been selected.

As a last step, in order to counter-balance the bad effect of the pruning, we consider some simple geometrical transformations of the patches (see Figure 2.12). These transformations are intended to enrich the dictionary with substantial variations in the patch structures, without giving out the coherence property (if two patches are neighbors, their transformed versions will remain neighbors).

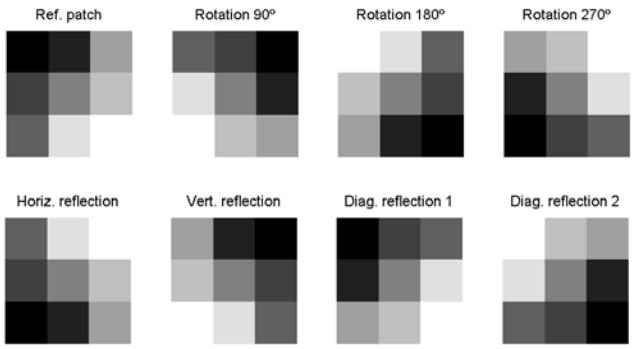


Figure 2.12: Geometrical transformations applied to the patches - The transformations considered include 3 rotations (for multiples of right angle) and 4 symmetrical reflections.

The prototypes and their transformed versions are finally used to form a new dictionary, whose size is about $8Mk$ pairs of patches ("about" because not all clusters may have at least M elements). Once the partitioning of the initial dictionary into K clusters is done, the value of M is then chosen according to the desired size of the final dictionary.

The so-designed dictionary learning process can be summarized in the following steps:

1. Take as input a large "multi-content" dictionary,
2. Perform *JKC* on this large dictionary,
3. Keep only M prototypes per cluster, and
4. Apply the geometrical transformations to the prototypes.

2.3.2 Experimental analysis

To test the dictionary construction strategy described in Section 2.3.1, we start from a large dictionary, concatenation of the 5 dictionaries mentioned in Table 2.5. The size of each single dictionary is 56000 pairs of patches; therefore, the size of the big one is 280000 pairs. The large dictionary is firstly used as an input of the *JKC* algorithm with $k = 750$. Some information about the convergence of the *JKC* clustering procedure applied to the large dictionary, e.g. the cost error value (intended as the difference between the current centers and those ones at the previous iteration), as well as some other statistics, are reported in Table 2.6.

Iteration	**Cost error**	**# non-assigned**
1	0.006285	205412
2	0.000357	127718
5	0.000056	95007
25	0.000008	86305
50	0.000004	84657
113	0	83655
% of placed vectors		70.1%
Min cluster size		1
Max cluster size		53270
Avg cluster size		261.8

Table 2.6: Statistics about one run of the *JKC* algorithm - The algorithm is applied to a dictionary of 280K pairs; the prospect of its convergence is reported, as well as other statistics.

As we can see from Table 2.6, the algorithm converges exactly to a solution after 113 iterations. The number of non-assigned patches (those ones in the "trash cluster") decreases progressively.

Once the big dictionary has been clustered, the patch selection phase is started: only $M = 12$ prototypes per clusters are taken, according to the selection strategy described in Section 2.3.1.2. The explained geometrical transformations are also applied. M is chosen such that the size of the final dictionary is comparable to the one of the starting dictionaries (i.e. around 56000 pairs), so achieving a reduction of the size of the big dictionary of about $\frac{1}{5}$. We run then the *NoNNE*

super-resolution algorithm presented in Section 2.2.3, for several images and two different factors (3 and 4). The results for all the dictionaries are reported in Table 2.7, where "BIG" stands for the concatenated dictionary and "FINAL" for the final dictionary we learn from it with our method (*JKC* + prototype selection).

		Dictionaries						
Image	**Scale**	**ESA**	**Flower**	**Heads**	**Synth**	**Wiki**	**BIG**	**FINAL**
Bird	3	32.13	**32.42**	32.17	31.21	31.88	32.27	32.31
Butterfly	3	24.89	24.22	23.77	23.95	24.37	24.94	**25.06**
Eyetest	3	18.99	18.64	18.58	**20.58**	18.62	20.48	20.55
Head	3	31.86	**32.07**	31.92	31.73	31.74	31.96	31.75
Newspaper	3	23.79	23.55	23.73	23.48	23.35	23.78	**23.85**
Bird	4	29.51	29.87	29.76	28.81	29.12	29.51	**29.90**
Butterfly	4	22.45	22.00	21.70	21.57	21.98	**22.48**	22.46
Eyetest	4	17.52	17.33	17.20	18.72	17.13	18.74	**18.94**
Head	4	30.55	**30.85**	30.69	30.35	30.42	30.70	30.82
Newspaper	4	21.93	21.92	22.01	21.88	21.68	22.02	**22.06**

Table 2.7: *PSNR* results of the *NoNNE* algorithm with the new designed algorithm - The final results are compared with those ones obtained with each of the single 5 input dictionaries, and with the dictionary "BIG", concatenation of them.

As we can see from Table 2.7, the new constructed dictionary performs better than any input dictionary for almost all images, while the size being comparable, confirming its goodness as a general-purpose dictionary. A significant case is represented by the "Eyetest" image, to which only one of the input dictionary is suitable ("Synth"): that means that a really particular image content is required to super-resolve it. Our designed dictionary succeeds also in this case. The good results are confirmed by Figure 2.13, where for 8 images (magnified by 4) we report the results of the worst and best dictionaries, according to each particular image, and our trained dictionary: the latter is always able to match the best performance, even if the best dictionary is different for each test image.

In comparison with the big dictionary, our constructed dictionary generally gives even better results, while having a markedly reduced size. The evolution of the *PSNR*, by considering all the steps between the big dictionary and the one finally constructed is reported in Table 2.8. The table shows the intermediate results after the first pruning step (removal of the trash cluster), and after the sampling of M prototypes per cluster (i.e. before the geometrical transformations are applied). The table also shows the results when our dictionary construction strategy is applied to a dictionary clustered via a traditional k-means procedure applied on the *concatenated* LR-HR vectors, instead of *JKC*.

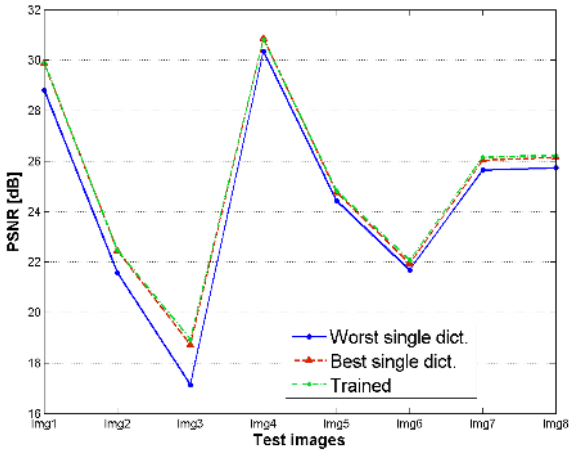


Figure 2.13: Best-, worst-case scenario, and our new constructed dictionary - The performance of our constructed dictionary is presented w.r.t. the best and worst single dictionary for any test image considered.

			JKC			k-means	
Image	**Scale**	**BIG**	**Prun.**	**Sampl.**	**FINAL**	**Sampl.**	**FINAL**
dict. size →		280000	196345	7060	56480	7139	58552
Bird	4	29.51	29.60	29.35	29.90	29.68	29.72
Butterfly	4	22.48	22.34	21.99	22.46	21.75	21.74
Eyetest	4	18.74	18.61	18.22	18.94	17.31	17.26
Head	4	30.70	30.71	30.68	30.82	30.75	30.83
Newspaper	4	22.02	21.95	21.96	22.06	21.96	22.06

Table 2.8: Evolution of the *PSNR*, for *JKC* and standard k-means - The evolution takes into account all the steps from the big dictionary to the one finally constructed; standard k-means is considered applied to the concatenated LR-HR vectors.

By looking at Table 2.8, we can see that, in the case of *JKC*, the pruning step usually brings only a slight decrease of the performance. A bigger, but still not dramatic, decrease is brought by the prototype sampling step, as the dictionary size drops severely. This effect is positively counterbalanced by the application of the geometrical transformations, that leads to the final dictionary. When comparing *JKC* to the k-means clustering applied to the concatenated vectors, we observe appreciably better results, so proving the effectiveness of the jointly clustering procedure.

2.3.3 Visual results with NoNNE and the newly-built dictionary

After the analysis conducted in Section 2.3.2, in this section we briefly look at the visual results of the output images super-resolved with the *NoNNE* algorithm.

To this end, Figure 2.14 reports the results related to the "Bird", "Buttefly", and "Head" images, all magnified by a factor of 3. The *NoNNE* algorithm is compared with the pyramid-based algorithm of [58], which is considered at the top of the state-of-the-art and uses internal information (i.e. self-similarities found in a pyramid of recursively scaled images) instead of an external dictionary. For [58], we report the outputs obtained thanks to a third-party implementation of the algorithm. The *NoNNE* algorithm, instead, is tested with one of the input dictionaries of Table 2.7 ("Esa") and the final dictionary constructed via the proposed strategy.

As we can see from the figure, the use of the new dictionary appreciably improves the visual results of the *NoNNE* algorithm (e.g. see the bamboo cane in the bird image). This makes it get closer to the performance of the pyramid-based algorithm of [58], which presents generally less blurred but somehow unnatural images. On the other hand, the algorithm in [58] requires much higher computational times (see Table 2.4).

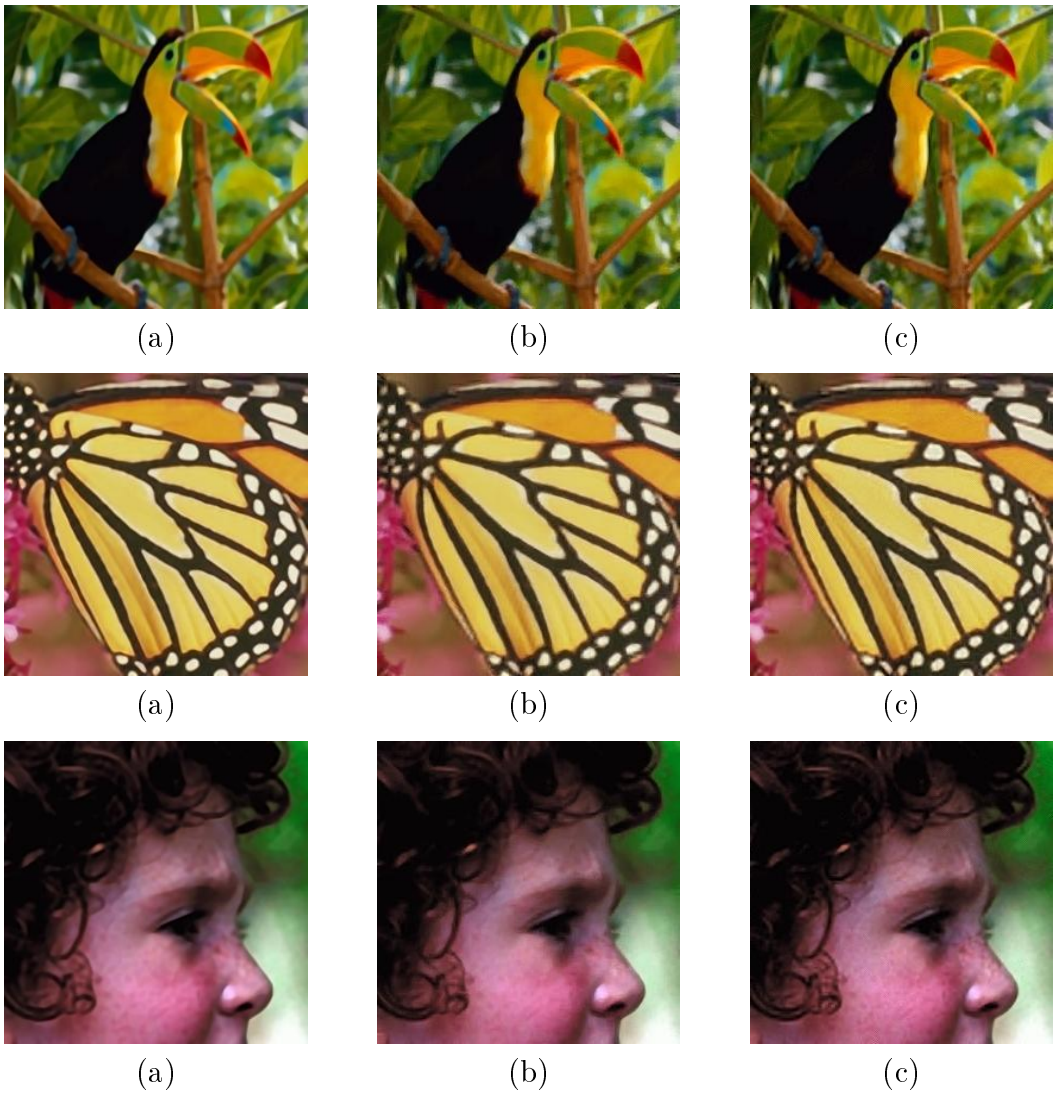


Figure 2.14: Visual comparisons between the *NoNNE* algorithm w/ or w/o the new dictionary and the state of the art - For each of the 3 test images we have results from: the pyramid-based algorithm of [58] (a), the *NoNNE* algorithm with the "Esa" dictionary (b), and the *NoNNE* algorithm with the final dictionary (c).

2.4 Enhanced interpolation and new patch generation schemes

In this section we present novel ideas that can be used in the framework of neighbor embedding (NE) based super-resolution. In particular, in Section 2.4.1 we introduce a new interpolation method that exploits the iterative back-projection (IBP) algorithm in a two-step procedure. We call this new method "enhanced interpolation", as it can be applied to any interpolation method, e.g. bilinear or bicubic interpolation, improving its result. By making use of the proposed enhanced interpolation and the concept of high-frequency residuals, we then propose in Section 2.4.2 new generation schemes for training LR and HR patches to compose an external dictionary for example-based SR.

Thanks to these elements (enhanced interpolation and new patch generation schemes), we can derive a new NE-based single-image super-resolution algorithm. The algorithm, which we synthetically call "NEEB", is fully described in Section 2.4.3. It is a single-image SR algorithm based on nonnegative neighbor embedding, as the *NoNNE* algorithm presented in Section 2.2. Moreover, it also makes use of the *JKC*-based dictionary construction procedure illustrated in Section 2.3.1. Therefore, the new *NEEB* algorithm can be seen as the evolution of the NE-based SR algorithms presented in this chapter, whose results specifically appeared in sections 2.2.4 and 2.3.3.

The work described in this section of the manuscript is present in one published paper of ours [82].

2.4.1 IBP and enhanced interpolation

An additional operation, performed by several SR algorithms (e.g. [58, 52]), once the output super-resolved image $\hat{I}_H$ is generated, consists in "back-projecting" the obtained image. This insures $\hat{I}_H$ to be consistent with the LR input image I_L. $\hat{I}_H$ is then corrected in an iterative fashion, by considering the error between the back-projected LR image $\hat{I}_L = \left(\hat{I}_H * b\right) \downarrow_m$ and the original LR image I_L. The update equation for this iterative method, which takes the name *iterative back-projection*, is

$$\hat{I}_H^{t+1} = \hat{I}_H^t + \left(\left(I_L - \hat{I}_L^t\right) \uparrow_m\right) * p \,, \tag{2.10}$$

where the LR image at iteration t $\hat{I}_L^t$ is obtained by back-projecting the related HR image $\hat{I}_H^t$, and p is a back-projection filter that locally spreads the differential error. IBP can have a fairly important role in example-based algorithms, as our *NoNNE* algorithms, where the HR output image is obtained by averaging reconstructed HR patches. While this averaging operation inevitably smooths

out certain HR details, IBP can then be effective in re-sharpening areas with edges and detailed texture.

As said, in SR algorithms IBP is often used at the very last stage. However, when the image generation process 1.8 is assumed to be known, and in particular the nature of the Blur kernel underlying it, this information can be already used at the beginning of the algorithm, in order to start with a "better guess" of the HR output image to be estimated. Let X be a LR input image and $\mathcal{I}(X)$ an interpolated version of it at the desired HR size, obtained with any traditional analytic interpolation method. $\mathcal{I}(X)$ represents a "rough" estimation of the unknown HR image. However, we believe that $\mathcal{I}(X)$ can be further improved, by applying the IBP algorithm, i.e. by back-projecting it to the LR dimension and consequently correcting it, once the differential error has been "forth-projected". We call this interpolation correction via IBP "enhanced interpolation", the new image being indicated by $\mathcal{E}(X)$.

Our intuition can be easily proved by testing the method with an input image. Figure 2.15 shows the results ("normal" and enhanced interpolation) for a given test image.

Figure 2.15: "Normal" and enhanced interpolation of a LR input image - The figure shows the evolution from X, a given LR input image, to its interpolated versions $\mathcal{I}(X)$ and $\mathcal{E}(X)$.

As we can observe from Figure 2.15, the enhanced-interpolated image $\mathcal{E}(X)$ has a definitely better look, as it represents a deblurred and sharpened version of the initially interpolation $\mathcal{I}(X)$. In the next section, the concept of enhanced interpolation is exploited also in the training phase, when generating the dictionary of patches.

2.4.2 New patch generation schemes

In example-based SR an important aspect is represented by how the training set is generated. In the case of external dictionaries, we have a HR external image J_H, we generate its LR counterpart J_L by following image generation model itself

(1.8) (i.e $J_L = (J_H * b) \downarrow_m$), and from them, or processed version of them, we extract the training patch pairs. Precisely, we extract:

- From the processed J_H, HR patches to be used in the HR output image reconstruction;
- From the processed J_L, LR patches to match the patches from the LR input image.

As a first scheme (see Figure 2.16), as done in [52], we can use directly J_H as a source for HR and the bicubic interpolation of J_L, $\mathcal{I}(J_L)$, as a source for LR patches. Therefore, since the two source images have equal size, we can sample same-size patches (e.g. 5×5 patches) exactly at the same locations. Note that here the acronyms LR and HR assume a wider meaning, as they do not refer to actual differences in the pixel resolution (the patches have the same size). It would then be more appropriate to talk about low-frequency and high-frequency contents of the patches, but for reasons of uniformity with the rest of the manuscript we decide to keep the usual notation.

In [52], the LR patches finally consist of gradient values computed on $\mathcal{I}(J_L)$, whereas the HR patches are mean-removed patches directly extracted from J_H (we just sample the patches and we subtract the mean value of each single patch to any pixel of it). We prefer, instead, to use mean-removed patches in both cases. In Section 2.2.1, in fact, we have shown that, in the neighbor embedding scheme, the double choice of mean-removed patches ("centered features") is preferable, as it is less complex (i.e. NN search of smaller vectors) and brings better results in terms of final *PSNR*.

The second training scheme (Figure 2.17) comes similar to the first one, but with a substantial difference: instead of taking the bicubic interpolation $\mathcal{I}(J_L)$ as a source for LR patches, we consider an *enhanced interpolation* $\mathcal{E}(J_L)$. This enhanced interpolation is the result of a back-projection operation, by taking the bicubic interpolation as a first upscaled image to be refined, exactly as explained in Section 2.4.1 for an hypothetical input image. $\mathcal{E}(J_L)$ does not contain high frequencies as J_H, but it represents a better upscaling of J_L than $\mathcal{I}(J_L)$, and, since LR patches will be similarly produced starting from the input image I_L, a better "starting point" for the SR process.

In [48] the patches used in the reconstruction are obtained by sampling the residual image between J_H and an interpolation of J_L (what we called $\mathcal{I}(J_L)$). The idea is to use and combine significant high frequencies, in order to bring to the input image I_L real high-frequency (HF) details: $(J_H - \mathcal{I}(J_L))$ is in fact the high-pass filtered version of J_H. Inspired by [48], we propose instead to use our enhanced interpolation $\mathcal{E}(J_L)$, obtained by bicubic interpolation and back-projection, and compute on that the high-frequency residual image. The new "back-projection residual" $(J_H - \mathcal{E}(J_L))$ is an even higher-pass filtered version of

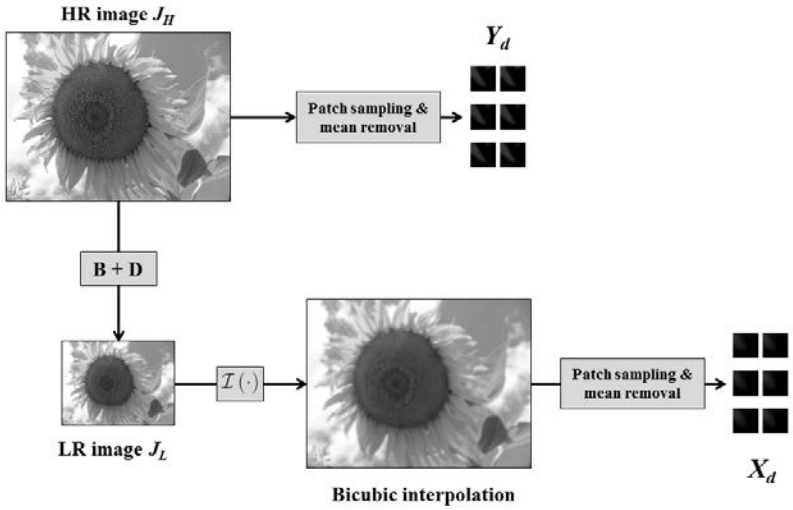


Figure 2.16: ***Scheme 1*** **for the generation of patches** - HR patches are sampled from a HR training image J_H; equally-sized LR patches are sampled from its interpolated LR version $\mathcal{I}(J_L)$. In both cases, centered features are used.

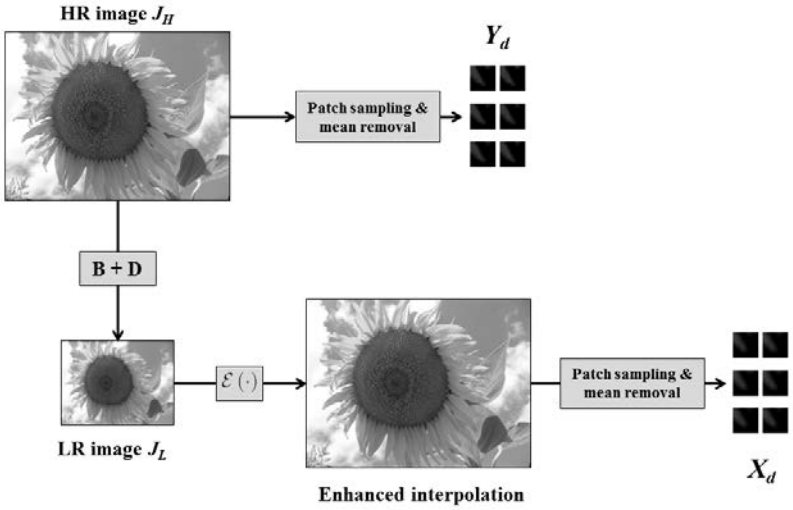


Figure 2.17: ***Scheme 2*** **for the generation of patches** - HR patches are sampled from a HR training image J_H; equally-sized LR patches are sampled from the *enhanced interpolation* of its interpolated LR version $\mathcal{E}(J_L)$. In both cases, centered features are used.

J_H, containing really substantial high frequencies. This third learning scheme is represented in Figure 2.18.

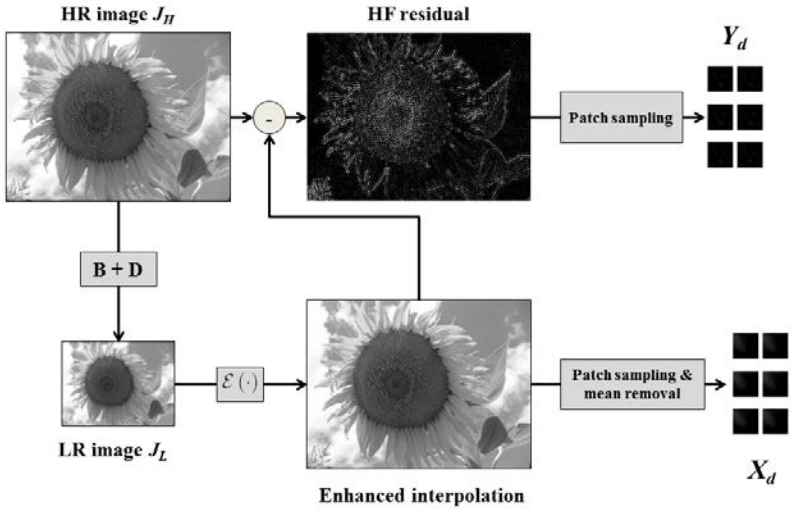


Figure 2.18: ***Scheme 3*** **for the generation of patches.** - LR patches are sampled from the *enhanced interpolation* of the LR training image $\mathcal{E}(J_L)$, and then represented by means of centered features. HR patches, with simple luminance values as features, are sampled from the high-frequency residual $J_H - \mathcal{E}(J_L)$.

Table 2.9 summarizes the three learning schemes described, specifying for each of them the sources for the LR patches and the HR patches.

	LR patch source	**HR patch source**
Scheme 1	$\mathcal{I}(J_L)$	J_H
Scheme 2	$\mathcal{E}(J_L)$	J_H
Scheme 3	$\mathcal{E}(J_L)$	$J_H - \mathcal{E}(J_L)$

Table 2.9: ***LR patches*** **and** ***HR patches*** **in the three learning schemes considered** - All patches are considered mean-removed, except for the back-projection residuals.

2.4.3 Neighbor Embedding SR algorithm using Enhanced Bicubic interpolation (NEEB)

In this section we define a new NE-based algorithm using the patch generation schemes described in Section 2.4.2, based on the enhanced interpolation method. We indicate this new algorithm with the acronym "NEEB" (Neighbor Embedding SR algorithm using Enhanced Interpolation). As the *NoNNE* algorithm of Section 2.2, we make use of a nonnegative neighbor embedding procedure. Moreover,

given the new dictionary construction procedure based on *JKC* described in Section 2.3.1, we want to test it in order to further improve the dictionary trained with the new schemes.

Thanks to all the "ingredients" presented so far, we are then able to design a complete algorithm for example-based SR. Initially, an external dictionary of LR patches and HR patches is formed by taking training images, alternatively according to one of the three training schemes described in Section 2.4.2. The dictionary so formed is then used as input of the dictionary construction process described in Section 2.3.1: the joint k-means clustering procedure helps improving the coherence of the dictionary at the level of local neighborhoods; then a sampling and enrichment strategy reduces the size of it. Finally, the SR task is achieved, by following the neighbor embedding scheme: for each input LR patch approximation, we look for the best nonnegative neighbor embedding, according to the *NNLS* minimization problem (2.7).

In Listing 2, the nonnegative neighbor embedding SR procedure for the training scheme 3 is reported. We assume that we already have a dictionary of LR patches, as mean-removed patches extracted from enhanced interpolations of LR training images $\mathcal{E}(J_L)$, and a dictionary of HR patches, as the corresponding high-frequency back-projection residual patches.

Listing 2 SR by nonnegative NE of back-projection residuals

procedure NEEB(I_L,$\mathcal{Y}^l$,$\mathcal{Y}^h$,K)
  $\mathcal{E}(I_L)$ ▷ **Enhanced interpolation of the LR input image**
  Divide $\mathcal{E}(I_L)$ into 5×5 patches (with a 4-pixel overlap)
  $\mathcal{X}^l = \{\mathbf{x}_i^l\}_{i=1}^{N_x}$ ▷ **Extract the LR input vectors**
  $\{\overline{x}_i^l\}_{i=1}^{N_x}$ ▷ **Store mean values**
  for $i \leftarrow 1, N_x$ **do**
    $\mathcal{N}_i = \underset{\{\mathbf{y}_k^l\}_{k=1}^{K} \in \mathcal{Y}^{l\,K}}{\arg\min} \sum_{k=1}^{K} \left\|\mathbf{x}_i^l - \mathbf{y}_k^l\right\|^2$ ▷ **Find K-NN in $\mathcal{Y}^l$**
    $\mathbf{Y}_i^l = \left[\mathbf{y}_1^l, \ldots, \mathbf{y}_K^l\right],\ \mathbf{y}_k^l \in \mathcal{N}_i$
    $\mathbf{w}_i = \arg\min_{\mathbf{w}} \|\mathbf{x}_i^l - \mathbf{Y}_i^l \mathbf{w}\|^2$ s.t. $\mathbf{w} \geq 0$ ▷ **Solve *NNLS***
    $\mathbf{x}_i^h = \mathbf{w}_i \mathbf{Y}_i^h$ ▷ **Compute the HR output vectors**
  end for
  $\mathbf{x}_i^h \leftarrow \mathbf{x}_i^l + \overline{x}_i^l + \mathbf{y}_t^i \quad i = 1, \ldots, N_x$ ▷ **Compute the actual HR patches**
  Combine together the HR patches to form the HR output image $\hat{I}_H$
  Iterative back projection to refine $\hat{I}_H$
end procedure

It is to be noticed that the input image I_L is first upscaled with the enhanced interpolation method described in Section 2.4.1 (step 2 of Algorithm 2). The

enhanced interpolation becomes therefore our starting point for the SR process, as the LR patches are extracted from it and the actual HR output patches are computed just by summing to them the averaged HF residual ($\mathbf{x}_i^h$). Algorithm 2 is easily adaptable to the other training schemes in Table 2.9, after slight modifications.

2.4.4 Analysis of the gain of each contribution

As a first test, we evaluate our nonnegative NE algorithm with the different training schemes reported in Table 2.9. In *Scheme 1* the LR patches are extracted from an interpolated version of the LR image $\mathcal{I}(I_L)$, whereas the HR patches are mean-removed patches taken from the HR image I_H. In *Scheme 2*, w.r.t. *Scheme 1*, we just replace the bicubic interpolation with our enhanced interpolation, achieved thanks to an extra IBP step, $\mathcal{E}(I_L)$. As for the IBP operation, we choose as back-projection filter p (see equation (2.10)) a Gaussian kernel of variance $\sigma^2 = 2$. *Scheme 3*, in turn, represents a modification of *Scheme 2*, where we take high-frequency residuals (sampled from $I_H - \mathcal{E}(I_L)$) as HR patches, instead of directly sampling the HR training images. As the three schemes represent each one a slight variation of the previous one, we can progressively evaluate the contribution of each feature introduced. To *Scheme 3*, we also add the *JSK*-based dictionary learning procedure described in 2.3.1, in order to improve the dictionary already formed and contextually reduce its size.

The results are reported in terms of *PSNR* of the super-resolved images, w.r.t. the ground-truth image. Each LR input image I_L is generated from the HR ground truth I_H, by blurring the latter with a Gaussian filter of variance $\sigma^2 = 1$ and downsizing it. As for the NE procedure, in all experiments we use a number of neighbors $K = 15$ and the neighbor embedding method described in Section 2.2.2.

Tables 2.10 and 2.11 report the results, under the scenarios described, for 5 images magnified by, respectively, a factor of 3 and a factor of 4.

As we can see from the tables, by passing from *Scheme 1* to *Scheme 2*, i.e. by introducing our enhanced interpolating method, we have an appreciable gain in terms of *PSNR* (between 0.1 and 0.4 dB). No big differences in terms of performance are, instead, between *Scheme 2* and *Scheme 3*: the use of HF back-projection residuals instead of mean-removed HR patches seems to work better for higher magnification factors (e.g. 4), but it leads to slightly worse performance in the case of a factor of 3.

A big gain comes instead, in most of the cases (especially for a scale factor of 3), from the *JKC*-based dictionary learning procedure described in [78], when applied to *Scheme 3*. The procedure, as explained in Section 2.3.1, consists initially of a joint k-means clustering of the LR patches and the HR patches; at

	Bird	Butterfly	Hat	Head	Lena
Scheme 1	32.66	25.30	29.27	32.21	29.98
Scheme 2	32.98	25.46	29.30	32.30	30.19
Scheme 3	32.85	25.37	29.26	32.26	30.13
Scheme 3 + JKC	33.37	26.36	29.65	32.28	30.52

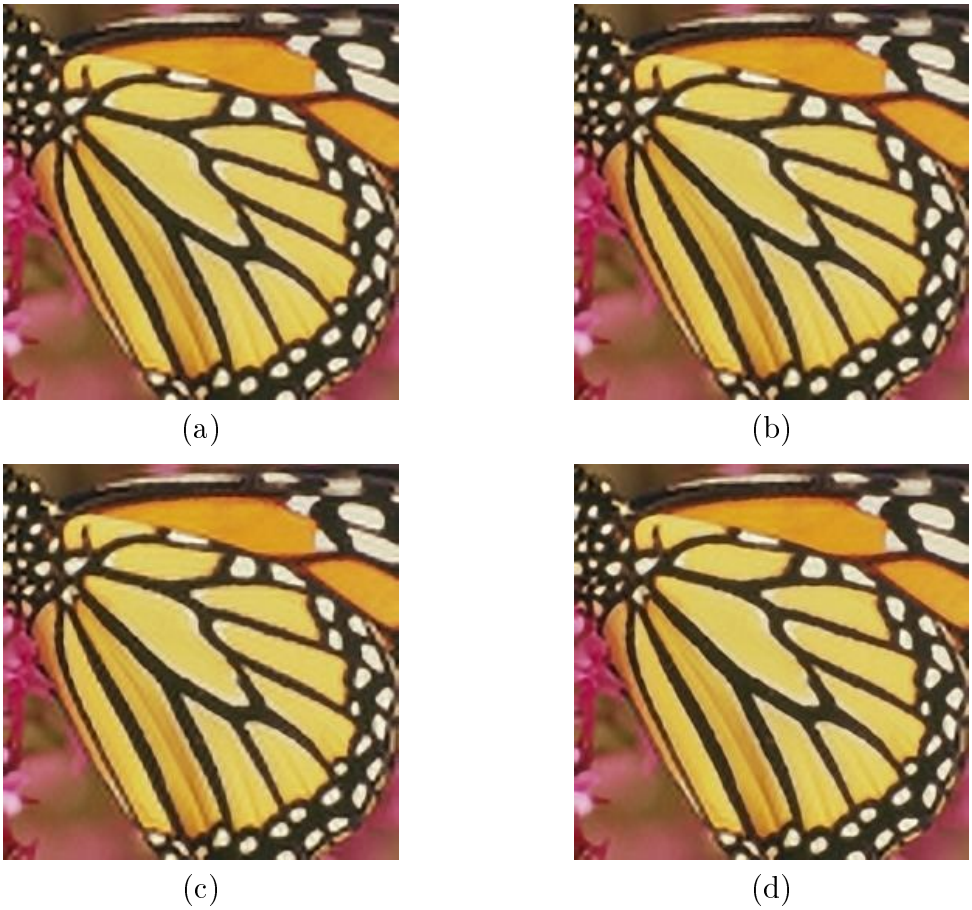
Table 2.10: Comparisons between the different training schemes, when upscaling with the *NEEB* algorithm by a factor of 3 - 5 different images are super-resolved; the 3 patch generation schemes described are compared (the *JKC*-based dictionary construction procedure is also performed on top of the last one).

	Bird	Butterfly	Hat	Head	Lena
Scheme 1	30.06	22.77	27.71	30.90	28.16
Scheme 2	30.41	23.03	27.80	31.05	28.41
Scheme 3	30.45	23.12	27.84	31.07	28.45
Scheme 3 + JKC	30.41	23.41	27.85	30.92	28.52

Table 2.11: Comparisons between the different training schemes, when upscaling with the *NEEB* algorithm by a factor of 4 - 5 different images are super-resolved; the 3 patch generation schemes described are compared (the *JKC*-based dictionary construction procedure is also performed on top of the last one).

the end of the clustering process, some pairs of patches may not find a placement, and so they are discarded. In this test we start from a dictionary of 500000 patches and try to cluster them into $K = 450$ classes. After the clustering process we have only 115324 patches placed into clusters, i.e. about the 23% of the initial number of patches, fairly below to the percentage presented in Table 2.6 (i.e. 70.1%). We explain this with the fact that we perform the clustering on high-frequency residuals, which are less correlated to the respective low-frequency patches than the HR "full-spectrum" patches used in the tests of Section 2.3.2. Therefore, more diverging assignments occur, and the pairs of patches that finally remain are particularly significant. After *JKC*, a patch sampling procedure and a dictionary enrichment with geometric transformations are performed, as described in Section 2.3.1.2. The final size of the dictionary is about 50000 pairs (1/10 of the original one).

In Figure 2.19, super-resolved images, related to the methods in Table 2.10, are reported, for the "Butterfly" image magnified by a factor of 3. The fourth solution (*Scheme 3 + JSK*-based dictionary learning) is clearly the one presenting the most pleasant visual result (e.g. see the dark stripes on the butterfly wings), so justifying the 1 dB gain.



(a) (b) (c) (d)

Figure 2.19: Visual comparisons on the butterfly image between the different patch generation schemes of the *NEEB* algorithm - The image is super-resolve by a factor of 3 with the following configurations: (a) *Scheme 1* (b) *Scheme 2* (c) *Scheme 3* (d) *Scheme 3* + *JSK*-based dictionary learning.

2.4.5 Results with NEEB

In Section 2.4.4 we showed that *Scheme 3* (the enhanced interpolation of the LR image and the back-projection residual used as sources of, respectively, LR patches and HR patches), followed by the *JKC*-based dictionary learning procedure, gives convincing results.

In this section we compare then our proposed algorithm with other single-image SR algorithms. In particular, we consider the SR algorithm via sparse representation of Yang *et al.* [52], our nonnegative neighbor embedding algorithm with centered features (*NoNNE*) presented in Section 2.2, and the pyramid-like algorithm of Glasner *et al.* [58] (provided by a third-party implementation).

The results, in terms of *PSNR* of the super-resolved image, for 5 images and

scale factors of 3 and 4, are reported in Table 2.12.

Image	**Scale**	**Sparse SR**	**NN-NE**	**Pyramid-like**	**Proposed**
Bird	3	32.91	32.47	32.96	**33.37**
Butterfly	3	24.92	25.27	**26.38**	26.36
Hat	3	29.20	29.34	29.47	**29.65**
Head	3	32.21	32.03	32.06	**32.28**
Lena	3	30.05	30.03	30.14	**30.52**
Bird	4	29.98	29.72	30.37	**30.41**
Butterfly	4	22.18	22.64	**24.41**	23.41
Hat	4	27.31	27.66	**28.44**	27.85
Head	4	30.83	30.66	**31.02**	30.92
Lena	4	27.97	28.07	**28.79**	28.52

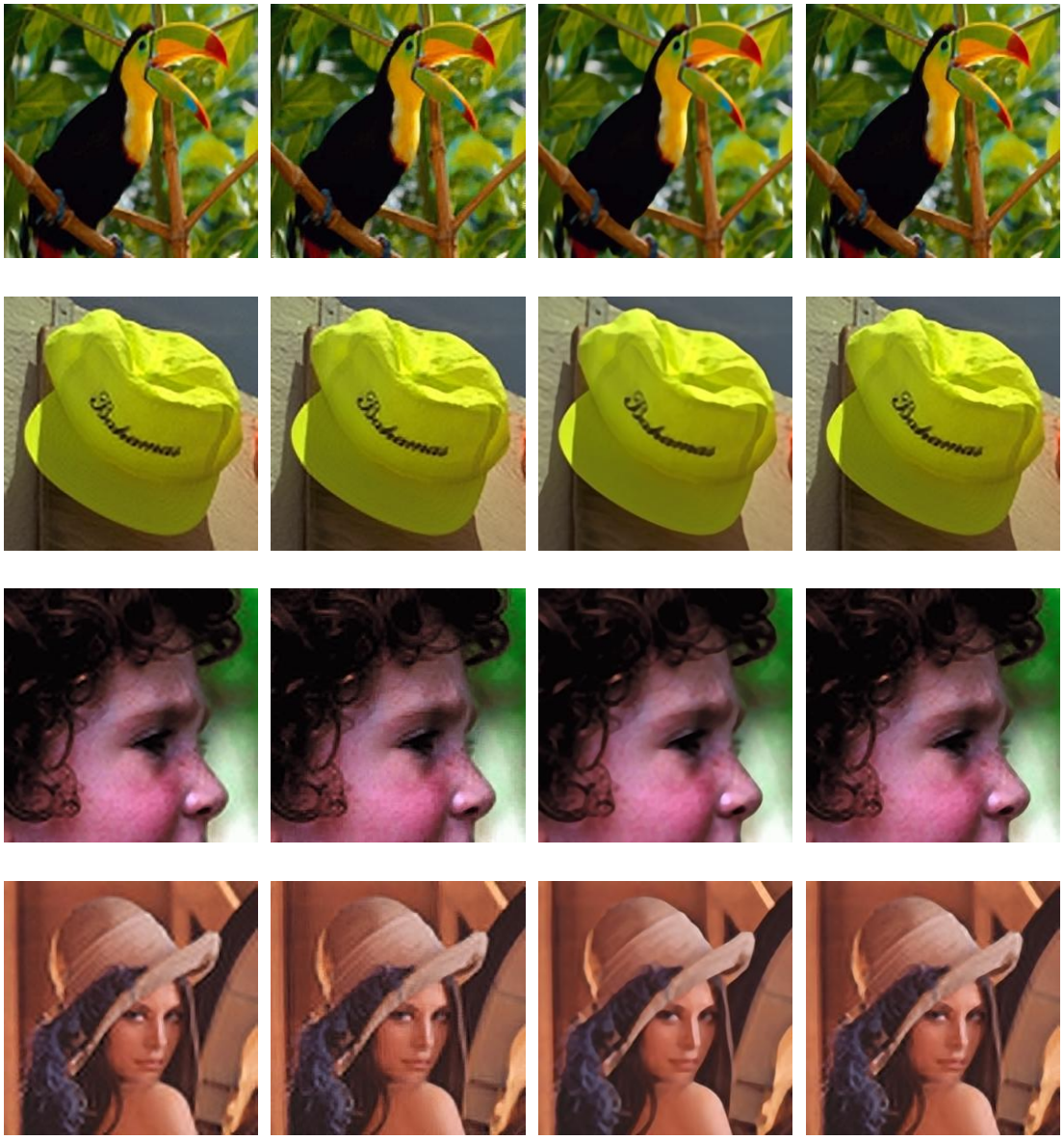
Table 2.12: Comparative results of the *NEEB* algorithms with the state of the art - *PSNR* values of the super-resolved image are provided for our proposed method and three methods in the literature.

By looking at the values of the table, our proposed method turns out to be the best one in half the cases (5 out of 10). It always outperforms the sparse method of Yang *et al.* [52] and our previous *NoNNE* algorithm, with respect to which it represents a substantial improvement. Moreover, it shows to be highly competitive with the method of Glasner *et al.* [58], which requires the SR procedure to be repeated several times by smaller magnifications.

The visual results, some of which are reported in Figure 2.20, confirm the good quantitative outcome. In particular, our method appears to be good in avoiding artifacts, while presenting natural and pleasant results (e.g. see the beak of the bird or the or the overall look of Lena).

To sum up, in this section we presented a new algorithm for single-image SR, which we referred to as *NEEB*, based on external dictionary and nonnegative neighbor embedding as our *NoNNE* algorithm previously presented. In comparison with *NoNNE*, the *NEEB* algorithm takes advantage of new schemes for the patch generation process, which are based on an enhanced interpolation procedure (the given image is first upscaled and then refined thanks to IBP). The underlying idea is to use IBP, typically employed at the end of the SR process, also at the beginning. This has two advantages. First, we start the SR algorithm with a better guess of the HR image. Second, we can have patches containing very salient high frequencies, by sampling them from the residual between each HR training image and the related enhanced-interpolated LR image. The *JKC*-based dictionary construction procedure, presented in Section 2.3, is also employed to subsequently optimize the dictionary.

The *NEEB* algorithm that derives from these new considerations has been



Yang *et al.* **NoNNE** **Glasner *et al.*** **NEEB**

Figure 2.20: Visual comparisons between the *NEEB* algorithm and other 3 methods in the literature - The images considered are: *bird* $\times 3$, *hat* $\times 3$, *head* $\times 3$, and *lena* $\times 4$.

compared with the *NoNNE* algorithms and other single-image SR methods. The results are very encouraging: it outperforms, both visually and in terms of quantitative results, *NoNNE* and other one-pass algorithms using external dictionaries (e.g. the sparse SR algorithm of Yang *et al.*), and it also does better than the

pyramid-based SR algorithm of Glasner *et al.* [58] in 5 cases out of 10, while presenting similarly acceptable visual results. The pyramid-based algorithm, however, as it progressively reconstruct the SR image in many passes, requires more computational time.

2.5 Conclusion

In this chapter we presented our work on example-based SR using an external dictionary. The employment of an external dictionary goes towards the direction of designing low-complexity algorithms, as the dictionary can be built in advance (no online operations are required) and possibly optimized for the SR problem.

As for the proper SR reconstruction procedure, we decided to adopt the *neighbor embedding* (NE) approach. Starting from the well-known LLE-based NE algorithm of Chang *et al.* [49], which paved the way to NE-based super-resolution, we proposed a new method based on nonnegative NE and chose for it the best feature representation to be adopted when representing the LR and HR patches.

After providing a general overview of NE in Section 2.1, each of the following sections presented a different contribution to the problem. In particular:

- In Section 2.2 we introduced the new nonnegative neighbor embedding method and proved the best efficiency in using mean-subtracted luminance values as patch features,
- In Section 2.3 we presented a new procedure for constructing a more compact a coherent dictionary based on a joint k-means clustering (JKC), and
- In Section 2.4 we proposed new training schemes to generate the patch data sets, based on the concept of "enhanced interpolation".

These three distinct contributions led to the formulation of different NE-based procedures. The three algorithms tested, by following each newly-presented contribution, are precisely:

- *NoNNE* (whose results are presented in Section 2.2.4),
- *NoNNE* with the JKC-based dictionary construction procedure (whose results are presented in Section 2.3.3), and
- *NEEB* (whose results are presented in Section 2.4.5).

The results reported in each related section proved the effective improvements brought by each distinct contribution. In particular, the *NEEB* algorithm, formerly presented in [82], shows the best visual and quantitative results. Compared to other state-of-the-art methods, it proves to be a very interesting algorithm, by generally outperforming other external-dictionary example-based methods, and presenting equally satisfying results w.r.t. more sophisticated methods employing internal dictionaries.

Chapter 3

Single-image SR based on linear mapping and internal dictionary

In this chapter we present a novel example-based SR algorithm, which is somehow "symmetrical" to the algorithms presented in Chapter 2. Different from them, which are based on the neighbor embedding (NE) approach, in fact, this algorithm falls into the category of direct mapping (DM) methods (see Section 1.2.2.1. Each HR output patch is the result of a mapping operation with a function formerly learned and directly applied to the corresponding LR input patch.

As for the other discriminating aspect in example-based SR, the typology of dictionary, here we explore the possibility of having what in Section 1.2.1 we called an "internal dictionary", the target being the maximization of the performance. In particular, the internal dictionary is built via a "double pyramid", where the traditional image pyramid of [58] is juxtaposed with a pyramid of interpolated images, and the DM-based reconstruction method consists in the learning of a linear mapping to be applied on the interpolated patches. The interpolation operation is done with the aim of making the computation of the single mapping functions via regression more robust (LR and HR patches, in fact, turn out to have the same sizes). Taking as a reference the well known algorithm of [58], the main contributions are then:

1. The modification of the training and upscaling scheme (i.e. the double pyramid), and
2. The employment of a DM method in the reconstructions, whereas in [58] the HR patches are reconstructed via NE.

Section 3.1, then, fully presents our new algorithm, by explaining how the internal dictionary is trained and the whole upscaling procedure. Section 3.2 instead reports some extensive experiments done: the different implementation

choices are here validated and the algorithm is compared with other state-of-the-art methods, by showing visual and quantitative results.

3.1 The "double-pyramid" algorithm

In this section we detail our novel SR algorithm, based on an internal dictionary of self-examples. Starting from the single image pyramid depicted in Figure 1.4, we propose a modified scheme with a "double pyramid" (Section 3.1.1). The self-examples found in this scheme are used to gradually upscale the LR input image up to the final super-resolved image, according to the cross-level scale factor chosen for the pyramid. The upscaling procedure employed falls within the local learning based reconstruction methods described in Section 1.2.2. In particular, it is a direct mapping method, where each LR patch is mapped into its HR version by means of a specifically learned linear function. The whole upscaling procedure and a summary of the whole algorithm are finally given in Section 3.1.2.

3.1.1 Dictionary construction: building the double pyramid

The goal of our SR algorithm is to retrieve the underlying HR image I_H from a degraded LR version I_L, which is supposed to be originated according to the image generation model (1.8). We choose the blur kernel B to be a Gaussian kernel with a given variance σ_B^2. The value s is instead an integer scale factor (e.g. 3 or 4), which is the factor by which we want the LR input image I_L to be magnified; i.e. if I_L is of size $N \times M$, the final super-resolved image $\hat{I}_H$ will have a size of $sN \times sM$.

For complexity reason, the SR algorithm later described is applied only on the luminance component Y of the input image I_L (a colorspace transformation from the RGB to the YIQ model is then possibly performed at the beginning), whereas the color components I and Q are simply upsized by Bicubic interpolation to the final desired size $sN \times sM$. In fact, since humans are more sensitive to changes in the brightness of the image rather than changes in color, it is a common belief that the SR procedure is worthy to be performed only on Y, so reducing the complexity of the algorithm by one third. Hereafter, then, all the image matrices and patch vectors must be intended as collections of pixel luminance values.

As a starting point for our internal dictionary learning procedure, we take the single pyramid depicted in Fig. 1.4. Here, the top-level is represented by the LR input image itself ($I_0 = I_L$). From it, a finite number of "sub-levels" is created, according to the following relation:

$$I_{-n} = (I_L * B_n) \downarrow_{p^n} , \tag{3.1}$$

where p, the pyramid cross-level scale factor, is typically a "small" number (e.g. $p = 1.25$). The sub-level image I_{-n} is then a particular rescaled version of the original image I_L (the total rescale factor amount to p^n). As for the variance of the Gaussian kernel B_n to which it is subjected, it can be computed according to the following formula, which is explained in [59]:

$$\sigma^2_{B_n} = n \cdot \sigma^2_B \cdot \log(p)/\log(s) \ . \tag{3.2}$$

Once the single pyramid is created, we propose now to interpolate each sub-level I_{-n} by the factor p. The so obtained interpolated level $\mathcal{U}(I_{-n})$, where $\mathcal{U}$ is an upscaling operator s.t. $\mathcal{U}(I) = (I) \uparrow_p$, is an image with the same size as the original non-interpolated level located just above in the pyramid I_{-n+1} (except for possible 1-pixel differences, due to the non-integer interpolation factors). We can then consider the pair constituted by $\mathcal{U}(I_{-n})$ and I_{-n+1} a pair of, respectively, LR and HR training images, from which derive a set of self-examples. By using all the pairs $\{\mathcal{U}(I_{-n}), I_{-n+1}\}$ for $n = 1, \dots NL$, where NL is the chosen number of sub-levels, and sampling at corresponding locations pairs of, respectively, LR and HR patches of equal size $\sqrt{D} \times \sqrt{D}$, we can then form our LR and HR internal dictionary sets: $\mathcal{Y}^l = \{\mathbf{y}^l_i \in \mathcal{R}^D\}_{i=1}^{N_y}$ and $\mathcal{Y}^h = \{\mathbf{y}^h_i \in \mathcal{R}^D\}_{i=1}^{N_y}$. Fig. 3.1 reports the scheme described, with the "double pyramid" formed by the traditional image cascade and, next to it, a side pyramid of interpolated levels.

3.1.2 Gradual upscalings

Once the LR and HR dictionary sets, $\mathcal{Y}^l$ and $\mathcal{Y}^h$, are formed, by populating them with the correspondences of self-examples found in the double pyramid described in Section 3.1.1, the proper SR reconstruction algorithm starts.

The algorithm consists in a multi-pass procedure, where the input image is gradually magnified by an upscale factor equal to the cross-level scale factor p. Given s as the total scale factor to be achieved, the number of necessary passes it then:

$$NP = \lceil \log_p s \rceil \ . \tag{3.3}$$

If s is not a power of p, the image super-resolved after NP passes will be over-sized w.r.t. to the targeted dimension ($sM \times sN$); which means that an extra resizing operation is needed. I_0 will be super-resolved into I_1, I_1 into I_2, and so on until obtaining I_{NP}, which will be possibly resized to obtain the SR estimated image $\hat{I}_H$ with the desired dimension. The multi-pass SR procedure is illustrated graphically in Fig. 3.2.

When generally upscaling the image I_n, this is first interpolated into the image $\mathcal{U}(I_N)$ from which a set of overlapping patches $\mathcal{X}^l = \{\mathbf{x}^l_i\}_{i=1}^{N_x}$ is formed, by scanning it with a sliding window of dimension $\sqrt{D} \times \sqrt{D}$. Each input patch $\mathbf{x}^l_i$ is

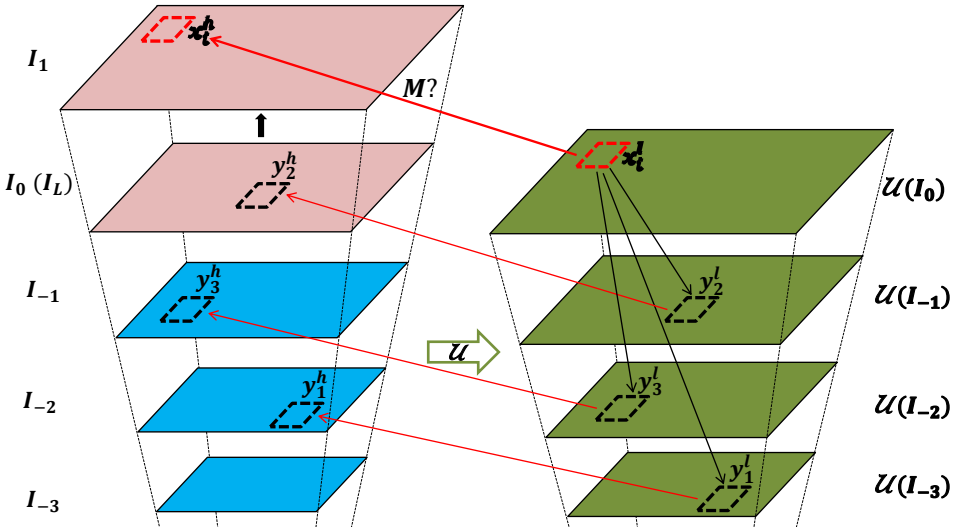


Figure 3.1: Creation of the "double pyramid" and search of self- examples throughout it - The figure concerns the upscaling of I_0 to I_1. Given a reference patch in the interpolated version of the starting image $\mathcal{U}(I_0)$, $\mathbf{x_i^l}$, 3 LR neighbors are found in the "side pyramid" of interpolated levels $(\mathbf{y_1^l}, \mathbf{y_2^l}, \mathbf{y_3^l})$. Thanks to these and the corresponding HR neighbors $(\mathbf{y_1^h}, \mathbf{y_2^h}, \mathbf{y_3^h})$, a linear function M is meant to be learned to directly map $\mathbf{x_i^l}$ into its corresponding HR output patch $\mathbf{x_i^h}$.

processed singularly and, after the learning based patch reconstruction procedure, a corresponding HR output patch $\mathbf{x}_i^h$ is produced. By iterating for all patches, we have at the end a set of HR reconstructed patches $\mathcal{X}^h = \{\mathbf{x}_i^h\}_{i=1}^{N_x}$, which will be finally re-assembled to form the upper-level image I_{n+1}. In the following paragraph we describe the patch reconstruction method adopted.

3.1.2.1 Direct mapping of the self-examples via multi-linear regression

While most of the "pyramid-based" SR algorithms in the literature [58, 59, 60] make use of a neighbor embedding (NE) based procedure to express each input and output patch in terms of combinations of self-examples, we follow instead the direct mapping (DM) approach. As explained in Section 1.2.2, DM aims at learning, thanks to the LR and HR local training sets, a mapping to directly derive the single HR output patch as a function of the related LR input patch.

DM has been employed in SR example-based algorithms using external dictionaries [65, 66], by using in particular the Kernel Ridge Regression (KRR) solution. In this case, the function space $\mathcal{H}$ is what is called a reproducing kernel Hilbert space (RKHS), and the single regression function f_i is seen as an expansion of

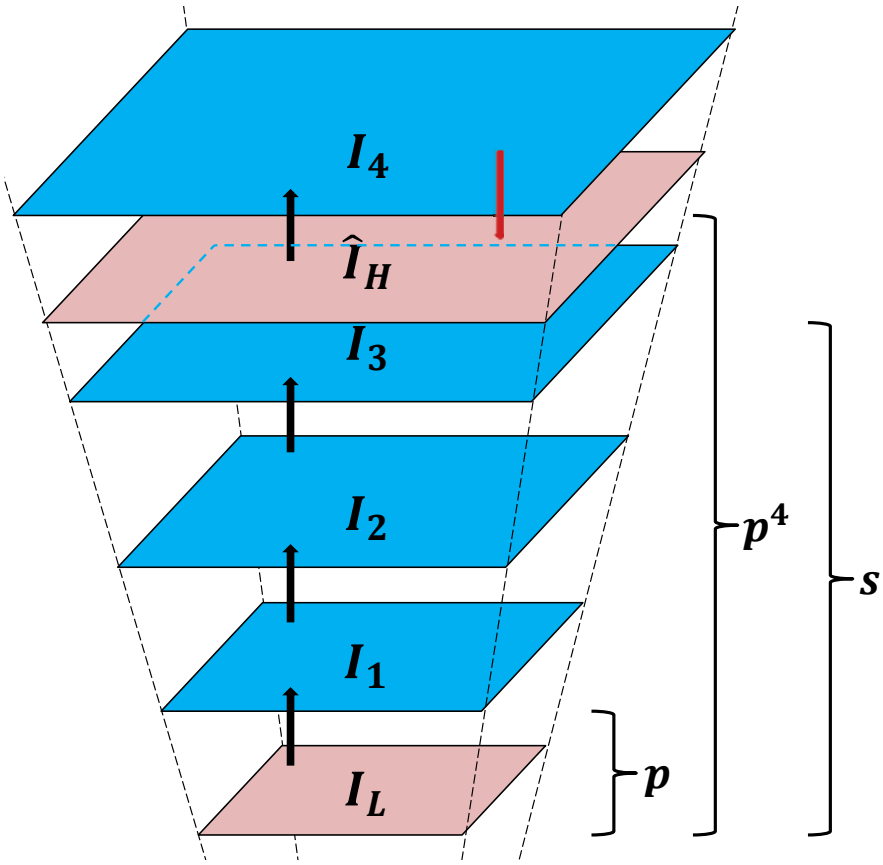


Figure 3.2: Estimation of the HR image by gradual upscalings - The original image I_L, by 4 upscalings by a factor of p, is super-resolved into I_4, that exceeds in dimension the desired scale factor s. $\hat{I}_H$ is obtained by a finally resizing operation.

kernel functions

$$f_i(\cdot) = \sum_{j=1}^{K} a_i(j)\mathsf{k}(\mathbf{y}_j, \cdot) \ . \tag{3.4}$$

Gaussian kernels of the kind $\mathsf{k}(\mathbf{x}, \mathbf{y}) = \exp\{-\sigma\|\mathbf{x} - \mathbf{y}\|_2^2\}$ are typically chosen.

For the sake of simplicity, we believe instead that, since in the case of internal learning the dictionary patches are more pertinent training examples, a simple linear mapping "can do the job". With this goal, $\mathcal{H}$ is taken as a linear function space

$$\mathcal{H} = \left\{ f(\mathbf{x}) = M\mathbf{x} \mid M \in \mathcal{R}^{D\times D}, \mathbf{x} \in \mathcal{R}^D \right\} \tag{3.5}$$

and the regularized empirical error (1.14) can be re-expressed as follows:

$$\begin{aligned} M_i &= \underset{M \in \mathcal{R}^{D \times D}}{\arg\min} \sum_{j=1}^{K} \|\mathbf{y}_j^h - M\mathbf{y}_j^l\|_2^2 + \lambda \|M\|_F^2 \\ &= \underset{M \in \mathcal{R}^{D \times D}}{\arg\min} \|Y_i^h - MY_i^l\|_2^2 + \lambda \|M\|_F^2 \,, \end{aligned} \tag{3.6}$$

where Y_i^l and Y_i^h are the usual, respectively LR and HR, local training sets related to the patch $\mathbf{x}_i^l$. In other words, we are looking for a linear transformation (i.e. the matrix M_i) to be directly applied to the LR input patch. This linear transformation is learned by observing the relations between the LR and HR dictionary patches which are neighbors with $\mathbf{x}_i^l$, according to the machine learning pattern and a regression model, where Y_i^l is the matrix of the *predictor variables* or regressors, and Y_i^h is the matrix of the *response variables*.

As the response variables are vectors and not scalars, we properly speak about *multi-variate regression* (MLR) [83]. The solution to (3.6) is known, and can be written in a closed-form formula:

$$M_i = Y_i^h {Y_i^l}^T \left(Y_i^l {Y_i^l}^T + \lambda I \right)^{-1} \tag{3.7}$$

where I is the identity matrix. The equation (3.7) corresponds exactly to what we called "model generation" (Step 2 of the local learning procedure described in Section 1.2.2). The prediction of the HR unknown patch (Step 3) is the straight application of the linear mapping learned:

$$\mathbf{x}_i^h = M_i \mathbf{x}_i^l \,. \tag{3.8}$$

Fig. 3.1 gives a rough depiction of how the DM reconstruction method works in the double pyramid.

3.1.2.2 Dictionary update

By learning for each input patch $\mathbf{x}_i^l \in \mathcal{X}^l$ a linear function M_i with the equation (3.7), and by applying this function to generate the related output patch, we end up with a collection of HR patches $\mathcal{X}^h = \{\mathbf{x}_i^h\}_{i=1}^{N_x}$ that need to be assembled to generate the current upscaling. Before the patch aggregation, the set of reconstructed HR patches $\mathcal{X}^h$ and the equivalent set of LR patches from which they have been originated, $\mathcal{X}^l$, are added to the dictionary as new correspondences of patches to be used in future upscalings: the patches of the two sets, in fact, are equal in number, and a LR-HR relation stands. The update of the dictionary is performed by simple set union, i.e. $\mathcal{Y}^l = \mathcal{Y}^l \cup \mathcal{X}^l$ and $\mathcal{Y}^h = \mathcal{Y}^h \cup \mathcal{X}^h$.

3.1.2.3 Patch aggregation and IBP

The patches of the input image, and consequently also in the equally-sized output image, are taken with a certain overlap; that means that, when they are re-placed in the original positions, at each pixel location we have a set of different candidate values. We can see the image at this stage as a 3-D image, where the multiple values per pixel are the results of different local observations of the input image (i.e. different patches taken), and therefore contain different partial information about the unknown HR image. We obtain a single output image by convexly combining these candidates at each pixel location: the convex combination is done by simply taking uniform weights, i.e. each candidate is weighted by $1/N_p$, where N_p is the number of overlapping patches contributing to that particular position.

After the image of the new level is formed, by overlapping and averaging, before it is used as a starting image for the next upscaling, it is further refined in an iterative fashion by the *iterative back-projection* (IBP) procedure. IBP is an additional operation adopted by several SR algorithms (e.g. [58, 52]), for which the output super-resolved image, once reconstructed, is "back-projected" to the LR dimension in order to assure it to be consistent with the LR input image, i.e. to assure it to be a plausible estimation of the underlying HR image, and conveniently corrected if errors are observed.

At iteration t of this refining procedure, the generic reconstructed n-th level I_n^t is first back-projected into an estimated LR image $\hat{I}_L^t$:

$$\hat{I}_L^t = \left(I_n^t * B_n\right) \downarrow_{p^n} \tag{3.9}$$

where B_n is a Gaussian blur with variance as expressed in (3.2). The deviation between this LR image found by back-projection and the original LR image is then used to further correct the HR estimated image:

$$I_n^{t+1} = I_n^t + \left(\left(I_L - \hat{I}_L^t\right) \uparrow_{p^n}\right) * b \,, \tag{3.10}$$

where b is a back-projection filter that locally spreads the differential error.

In Listing 3, our proposed SR procedure is described in a simplified manner, by reporting the pseudocode for the two main routines of the algorithm: "Internal Learning", where the double pyramid is constructed and the dictionary sets of LR and HR patches are initially formed, and "SingleUpscale", that reports the procedure to upscale a generic level I_n to the upper level. To be noted, in particular, on line 15 the *for* loop, which consists of 3 instructions and implements the 3 steps of the local learning based patch reconstruction procedure described in Section 1.2.2: nearest neighbor search, model generation, and prediction.

Listing 3 Single-image SR via linear mapping of interpolated self-examples

procedure INTERNALLEARNING(I_L, NL, s, p, σ_B^2)
  for $n \leftarrow 1, NL$ **do** ▷ **Create the pyramid levels**
    $\sigma_{B_n}^2 \leftarrow n \cdot \sigma_B^2 \cdot \log(p)/\log(s)$
    $I_{-n} \leftarrow (I_L * B_n) \downarrow_{p^n}$
    $\mathcal{U}(I_{-n}) \leftarrow (I_{-n}) \uparrow_p$
  end for

  for $n \leftarrow 1, NL$ **do** ▷ **Populate the internal dict.**
    Sample patches from $\mathcal{U}(I_{-n})$ and add to $\mathcal{Y}^l$
    Sample patches from I_{-n+1} and add to $\mathcal{Y}^h$
  end for
end procedure

procedure SINGLEUPSCALE(I_n, p, $\mathcal{Y}^l$, $\mathcal{Y}^h$)
  $\mathcal{U}(I_n) \leftarrow (I_n) \uparrow_p$ ▷ **Upscale the current level**
  Extract LR patches from $\mathcal{U}(I_n) \rightarrow \mathcal{X}^l = \{\mathbf{x}_i^l\}_{i=1}^{N_x}$

  for $i \leftarrow 1, N_x$ **do** ▷ **Single patch reconstructions**
    Find the local train. sets of $\mathbf{x}_i^l$ by NNS $\rightarrow Y_i^l, Y_i^h$
    $M_i \leftarrow Y_i^h Y_i^{l^T} \left(Y_i^l Y_i^{l^T} + \lambda I \right)^{-1}$
    $\mathbf{x}_i^h \leftarrow M_i \mathbf{x}_i^l$
  end for

  Form I_{n+1} with the constr. patches $\mathcal{X}^h = \{\mathbf{x}_i^h\}_{i=1}^{N_x}$
  Refine I_{n+1} by IBP
  $\mathcal{Y}^l \leftarrow \mathcal{Y}^l \cup \mathcal{X}^l$ ▷ **Update the LR dictionary**
  $\mathcal{Y}^h \leftarrow \mathcal{Y}^h \cup \mathcal{X}^h$ ▷ **Update the HR dictionary**
end procedure

3.2 Experimental Results

In this section we conduct some experiments on the single-image SR algorithm proposed in Section 3.1. In particular, in Section 3.2.1 we evaluate the different contributions, in terms of implementation choices, that led to its final formulation summarized in Listing 3. In Section 3.2.2, instead, we compare our algorithm with other state-of-the-art methods, by both showing visual comparisons on super-resolved images and reporting quantitative results, according to the *PSNR* and *SSIM* metrics. *PSNR* and *SSIM* values are obtained as measures of the distance between the HR original image, from which the LR input image, for test purposes, has been originated, and the super-resolved image. The image generation model adopted is the one expressed in Equation (1.8), with the variance of the Gaussian blur σ_B^2 set to 1 in all experiments. The interpolation method used is Bicubic interpolation.

As for the various parameters of the algorithms, they have been "tuned" by empirically looking for their optimal values. Notably, the cross-level scale factor p is taken as $p = 1.25$; as for the patch size, instead, 5×5 patches are sampled from the internal images with a 4-pixel overlap. For each input patch, then, $K = 12$ are selected in the dictionary via NNS.

3.2.1 Evaluation of the different contributions

In this section we want to assess the different contributions of our algorithm, which have been discussed in Section 3.1. With respect to the well-known single-image SR algorithm in [58], two main contributions have been presented:

1. The employment of a DM reconstruction method (i.e. MLR), instead of NE;
2. The introduction of a different training and upscaling scheme, i.e. the "double" pyramid.

To singularly evaluate the two "ingredients" above, we test three different procedures (summarized in Table 3.1), which all fall in the category of example-based single-image SR algorithms employing an internal dictionary.

We first start with an algorithm, where a single pyramid is constructed and NE with non-local means (NLM) weights (1.10) is used as patch reconstruction method ("Algorithm 1"). This algorithm is very close in the spirit to the method in [58], and thus its implementation can be considered as a reference for the mentioned method, except for possible slight differences in the code configuration and the choice of the parameters. With respect to Algorithm 1, "Algorithm 2" uses a DM patch reconstruction method instead of NE, i.e. the MLR method described in Section 3.1.2 that linearly maps each LR patch into the related HR patch. The finally proposed algorithm introduces then the double pyramid scheme, featuring the side pyramid of interpolated levels. Table 3.2 presents the

	Rec. Method	**Double pyr.**
Algorithm 1	NE	No
Algorithm 2	DM	No
Proposed	DM	Yes

Table 3.1: Summary of the internal-dictionary procedures considered - The different procedures are meant to evaluate each distinct contribution to our algorithm.

PSNR and *SSIM* values for all the considered algorithms, when tested on seven input images and for two different scale factors ($s = 3, 4$).

		Algorithm 1		**Algorithm 2**		**Proposed**	
Image	**Scale**	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*
Bike	3	23.13	0.792	23.20	0.796	23.30	0.799
Bird	3	33.71	0.884	34.06	0.890	34.13	0.890
Butterfly	3	27.08	0.833	27.38	0.840	27.56	0.841
Hat	3	29.95	0.651	30.11	0.655	30.11	0.655
Head	3	32.43	0.633	32.47	0.667	32.43	0.668
Lena	3	30.68	0.770	30.91	0.775	30.89	0.775
Woman	3	30.33	0.862	30.41	0.866	30.52	0.867
Avg gain w.r.t. A1		-	-	0.18	0.009	0.23	0.010
Bike	4	21.58	0.685	21.53	0.688	21.63	0.689
Bird	4	30.61	0.810	30.89	0.824	31.01	0.826
Butterfly	4	24.68	0.763	24.71	0.772	25.05	0.775
Hat	4	28.54	0.553	28.60	0.556	28.53	0.555
Head	4	31.17	0.556	31.28	0.591	31.23	0.590
Lena	4	29.02	0.681	29.22	0.687	29.28	0.689
Woman	4	27.60	0.778	27.90	0.793	27.96	0.793
Avg gain w.r.t. A1		-	-	0.13	0.012	0.21	0.013

Table 3.2: Performance results of the three different algorithms using an internal dictionary - The performance are measured in terms of *PSNR* and *SSIM* values, when super-resolving several images for factors of 3 and 4.

As we can observe from the table, from Algorithm 1 to the Proposed algorithm we have almost always a progressive consistent improvement in the SR

performance, with our finally proposed procedure presenting an average gain of about 0.22 dB w.r.t. to the traditional scheme based on a single pyramid and NE (Algorithm 1). This gain can be appreciated when observing the output images. Fig. 3.3 shows in fact the visual results obtained with the three different procedures, for two super-resolved images.

As we can observe from the zoomed-in areas of the images, with the finally proposed algorithm, we are able to produce finer details (see the branch of the tree behind the bird, or the texture of the hat of the woman). In particular, by observing the woman's hat, we can appreciate a sort of progress in the outcome of the three algorithms: Algorithm 1 gives a pretty blurred result; Algorithm 2, thanks to the use of DM in the place of NE, shows instead more regular structures; the edges and the shapes of these structures look even sharper with the Proposed algorithm, where the DM functions have been computed on the interpolated patches of the double pyramid.

3.2.2 Comparison with state-of-the-art algorithms

In this section we perform a comparative assessment of our method with other single-image SR algorithms in the state-of-the art, by extending the comparison also to SR methods based on external dictionaries. For this purpose, we consider Bicubic interpolation as a reference for traditional analytical interpolation methods, and four other example-based SR algorithms. The characteristic of the latter, as well as those ones of our proposed method, are summarized in Table 3.3.

Method	**Reference**	**Dictionary**	**Rec. Method**
LLE-based NE	Chang et al. [49]	External	NE with LLE weights (1.11)
Nonnegative NE	Bevilacqua et al. [82]	External	NE with NN weights (1.12)
Sparse SR	Yang et al. [52]	External	Sparse coding
Pyramid	Glasner et al. [58]	Internal, single pyramid	NE with NLM weights (1.10)
Proposed	**This paper**	**Internal, double pyr.**	**DM via MLR**

Table 3.3: Summary of the methods used for the comparison with our double-pyramid algorithm - Our method and four other example-based SR algorithms are considered in the comparison.

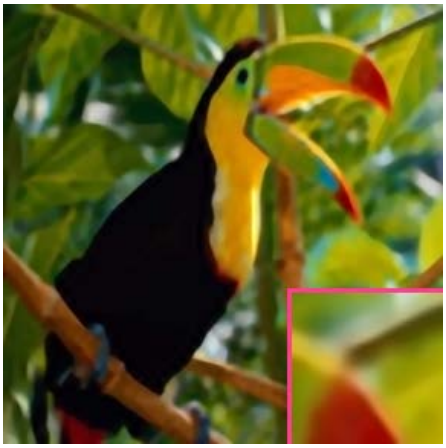
For the first three methods in the table, the original code of the respective authors, possibly slightly modified to make the comparison fair, has been used. For the "pyramid" method of [58], instead, the third-party code of the authors of [59] has been adopted. Table 3.4 reports the performance results of the algorithms considered (*PSNR* and *SSIM* values), with the usual set of seven test images, and 3 and 4 as scale factors.

		Bicubic		**LLE-based NE**		**NN NE**		**Sparse SR**		**Pyramid**		**Proposed**	
Image	**Scale**	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*
Bike	3	20.84	0.640	22.38	0.754	23.02	0.784	22.71	0.771	22.81	0.772	23.30	0.799
Bird	3	29.88	0.820	32.11	0.834	33.37	0.864	32.90	0.858	32.18	0.862	34.13	0.890
Butterfly	3	21.75	0.712	24.27	0.774	26.36	0.823	24.92	0.787	26.63	0.825	27.56	0.841
Hat	3	27.27	0.536	29.24	0.619	29.65	0.632	29.21	0.602	29.66	0.626	30.11	0.655
Head	3	31.02	0.583	32.05	0.626	32.28	0.638	32.23	0.653	31.69	0.629	32.43	0.668
Lena	3	28.03	0.670	29.78	0.729	30.52	0.754	30.06	0.738	30.28	0.748	30.89	0.775
Woman	3	26.17	0.778	28.27	0.806	29.35	0.840	29.02	0.815	29.89	0.857	30.52	0.867
Average		26.42	0.677	28.30	0.734	29.22	0.762	28.72	0.746	29.02	0.760	29.85	0.785
Bike	4	19.83	0.536	20.75	0.634	21.26	0.667	20.93	0.651	21.46	0.681	21.63	0.689
Bird	4	28.09	0.738	29.25	0.741	30.41	0.786	30.00	0.787	30.10	0.803	31.01	0.826
Butterfly	4	20.30	0.637	21.83	0.682	23.41	0.738	22.18	0.701	24.44	0.761	25.05	0.775
Hat	4	26.30	0.462	27.48	0.505	27.85	0.530	27.31	0.506	28.41	0.544	28.53	0.555
Head	4	30.07	0.516	30.71	0.537	30.92	0.556	30.85	0.573	30.86	0.575	31.23	0.590
Lena	4	26.85	0.593	27.84	0.628	28.52	0.670	27.98	0.649	28.70	0.668	29.28	0.689
Woman	4	24.61	0.697	25.84	0.696	26.76	0.752	26.18	0.732	27.20	0.777	27.96	0.793
Average		25.15	0.597	26.24	0.632	27.02	0.671	26.49	0.657	27.31	0.687	27.81	0.702

Table 3.4: Performance comparison between our double-pyramid algorithm and other state-of-the-art methods - The performance is measured in terms of *PSNR* and *SSIM* values, when super-resolving several images for scale factors of 3 and 4.

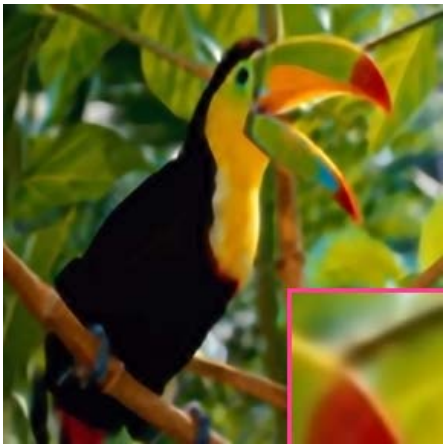
Table 3.4 shows clearly that our method outperforms the other algorithms in terms of objective quality of the super-resolved images, for all the images and the two scale factors considered. The results are confirmed when observing the visual comparisons (Fig. 3.4, Fig. 3.5, and Fig. 3.6).

From the visual results presented, we can see that methods based on external dictionaries ("LLE-based NE", "NN NE", and "Sparse SR") often present blurring and ringing artifacts (see e.g. the *Bike* images in Fig. 3.5). Results for the two algorithms based on an internal dictionary (the "pyramid" algorithm of [58] and ours), instead, are certainly more pleasant at sight, presenting sharp edges and smooth artifact-free results in the regions with no texture. However, the algorithm of [58] in some cases shows results that look somehow "artificial", with over-smoothed areas or unnatural edges (see Lena's eye in Fig. 3.6e, whose shape is deformed). Our algorithm succeeds also in avoiding these undesired effects.



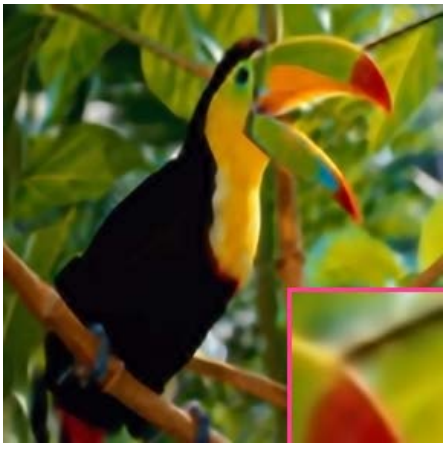
Algorithm 1

Algorithm 1



Algorithm 2

Algorithm 2



Proposed

Proposed

Figure 3.3: Super-resolved images with zoomed-in areas for the three different internal-dictionary procedures - The image and the scale factors considered are: *Bird* $\times 4$ (left) and *Woman* $\times 4$ (right).

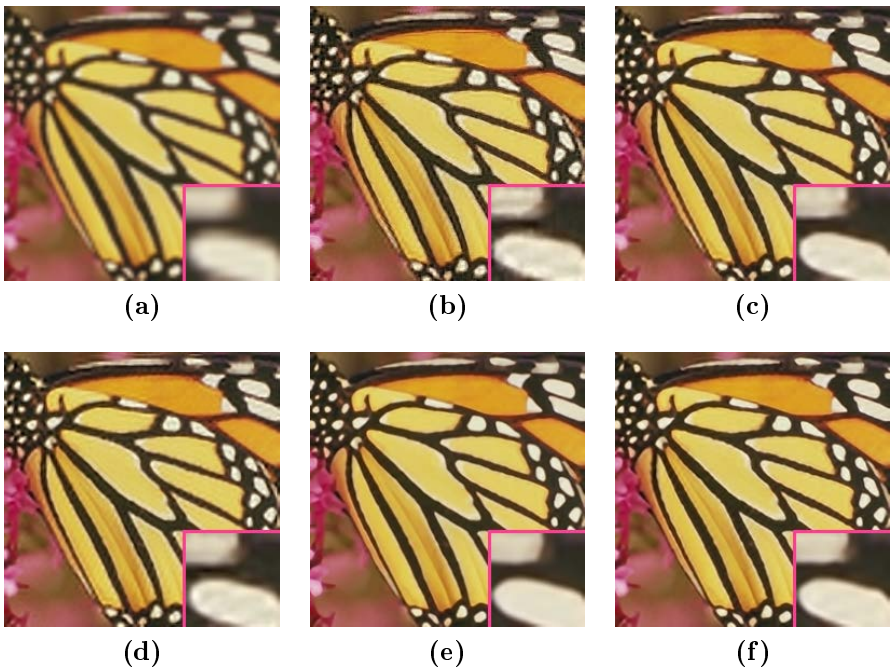


Figure 3.4: Comparative results with zoomed-in areas for *Butterfly* magnified by a factor of 3 - The methods considered are: (a) Bicubic interpolation, (b) LLE-based NE, (c) NN NE, (d) Sparse SR, (e) Pyramid, (f) Proposed.

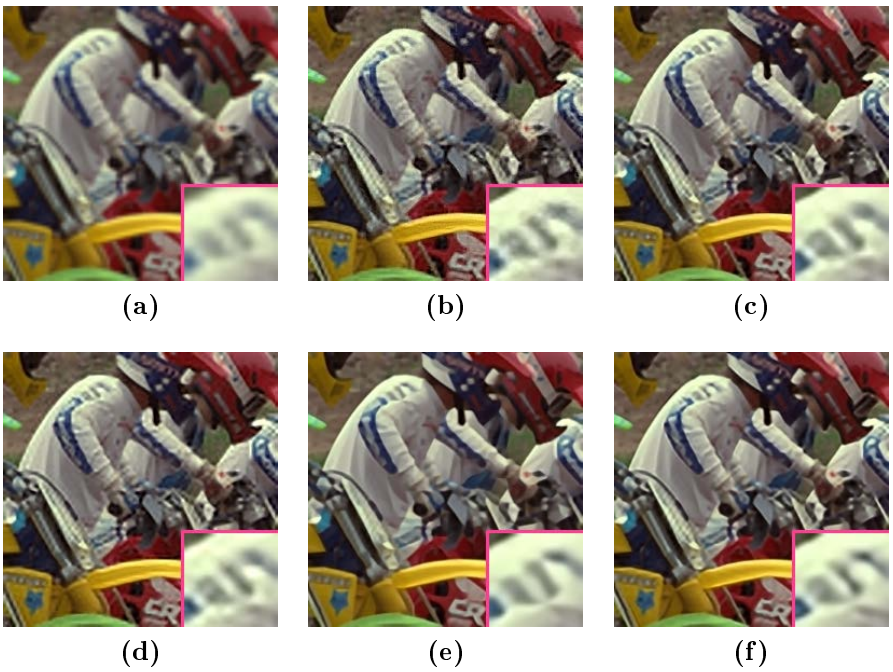


Figure 3.5: Comparative results with zoomed-in areas for *Bike* magnified by a factor of 3 - The methods considered are: (a) Bicubic interpolation, (b) LLE-based NE, (c) NN NE, (d) Sparse SR, (e) Pyramid, (f) Proposed.

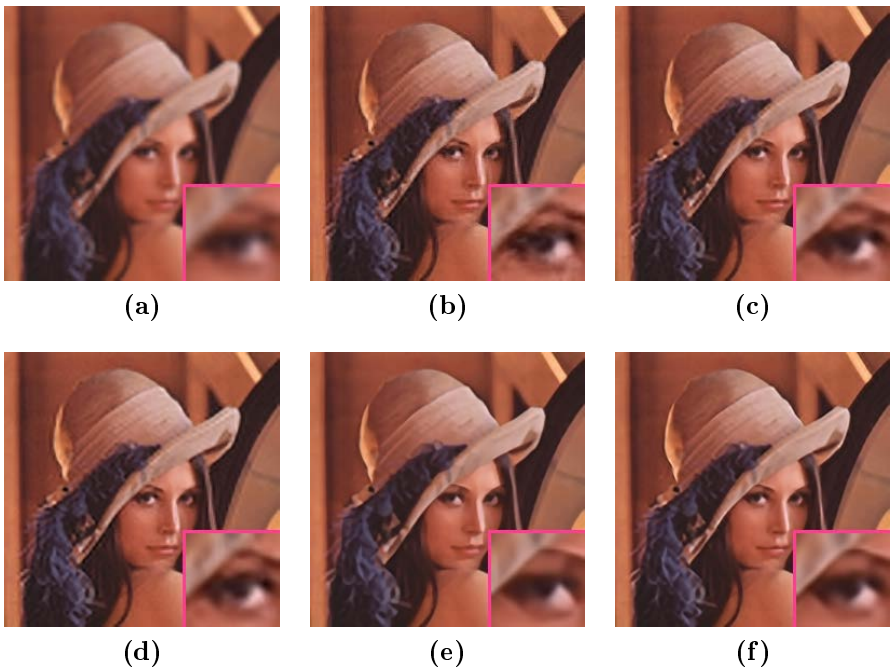


Figure 3.6: Comparative results with zoomed-in areas for *Lena* magnified by a factor of 3 - The methods considered are: (a) Bicubic interpolation, (b) LLE-based NE, (c) NN NE, (d) Sparse SR, (e) Pyramid, (f) Proposed.

3.3 Conclusion

In this chapter we presented a novel single-image SR method, which belongs to the family of example-based SR algorithms, using an internal dictionary of patches in the upscaling procedure. The algorithm originally makes use of a "double pyramid" of images, built starting from the input image itself, to extract the dictionary patches (thus called "self-examples"), and employs a regression-based method to directly map the low-resolution (LR) input patches into their related high-resolution (HR) output patches. When compared to other state-of-the-art algorithms, our proposed algorithm shows the best performance, both in terms of objective metrics and subjective visual results. As for the former, it presents considerable gains in *PSNR* and *SSIM* values. When observing the super-resolved images, also, it turns out to be the most capable in producing fine artifact-free HR details. The algorithm does not rely on extra information, since it uses an internal dictionary automatically "self-adapted" to the input image content. Moreover, it has few parameters that are easy to tune. This makes it a particularly attractive method for SR upscaling purposes, when no complexity constraint

Chapter 4

Super-resolution of a video sequence

In this chapter[1] we address the problem of extending to the video super-resolution case the theory and the results developed in the previous chapters of this manuscript (particularly, on example-based SR algorithms using either an external or an internal dictionary, respectively in Chapter 2 and Chapter 3).

According to the application-based classification we made in Section 1.1, video super-resolution (or *video-to-video* SR) corresponds to the third category of SR algorithms, i.e. MIMO (Multi-input multi-output) SR. As said, MIMO super-resolution is not directly related to a precise family of actual methodologies, but elements from *single-image* and *multi-frame* SR methods are alternatively exploited when trying to solve the problem of upscaling a video sequence. As in this doctorate we mostly focused on designing algorithms belonging to the single-image SR category, we want to adapt these algorithms to the video-to-video case. The video SR problem will then be addressed from the single-image point of view, where, basically, each frame is separately upscaled. Issues like the temporal consistency between the different upscaled frames, though, will still be taken into account.

When upscaling a video sequence, we can consider two scenarios:

1. **Scenario A**: The video sequence also contains certain HR key frames, appearing with a fixed frequency f_K (see Figure 4.1).
2. **Scenario B**: The video sequence contains only LR frames (see Figure 4.2).

The two SR problems related to the scenarios considered (*Scenario A* and *Scenario B*) are tackled, respectively, in Section 4.2 and Section 4.3. Before that, in Section 4.1 we introduce a new method for computing the patch reconstruction weights in the case of neighbor embedding SR, that, when upscaling a certain frame, takes into account also previously reconstructed frames. These weights

1. Part of the concepts and methods described in this chapter appeared in two publications [84, 85]

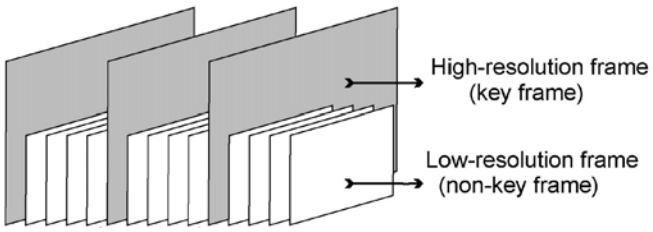


Figure 4.1: ***Scenario A*** **considered in the upscale of a video sequence**
- Other then LR frames, periodic HR key frames appear in the sequence.

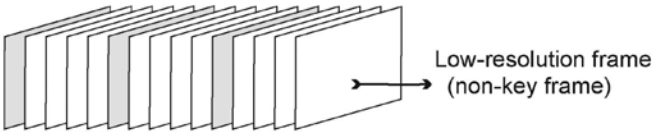


Figure 4.2: ***Scenario B*** **considered in the upscale of a video sequence**
- The sequence consists only of LR frames.

are intended to improve the temporal consistency between frames reconstructed with the single-image approach and can be used in both scenarios.

4.1 Weight computation method to provide temporal consistency

In order to upscale the LR frames, a trivial solution to this problem is to simply super-resolve each frame independently by means of a single-image SR algorithm, as done in [45]. We would, however, like to add to the framework a motion estimation step as in [86], in order to take advantage of the already reconstructed frames and enforce temporal consistency.

To this end, in Scenario A (Figure 4.1), we can use the original HR key frames as references for performing motion estimation. In Scenario B (Figure 4.2), instead, we do not have such frames: however, we can "create" a few key frames according to the same frequency f_K, by first super-resolving certain LR frames of the input video sequence, and use these frames to help the upscaling of the intermediate ones.

In the context of neighbor embedding (NE) super-resolution, whose general procedure is described in Section 2.1, a way to promote temporal consistency between frames, while still performing individual upscalings, could be to take into account also the neighbor key frames, every time the reconstruction weights for a given patch are computed. More precisely, when reconstructing the HR version of a certain LR input patch located in an intermediate frames, we could

search in the key frames the patches that, in their LR version, look the closest to it and drive the HR reconstructed patch to be "similar" to these patches. In other words, similar patches must lead to similar reconstructions along neighbor frames.

We then propose a new weight computation method for NE according to the principle described above, where the closest patches are found by motion estimation as in [86]. The motion estimation process is done for every overlapping patch: for a given LR input patch in a non-key frame $\mathbf{x}_i^l$, we can then find the two LR patches in the two neighbor key frames, pointed by the motion vectors found, $\mathbf{x}_{i,K1}^l$ and $\mathbf{x}_{i,K2}^l$, and, consequently, their corresponding HR versions, $\mathbf{x}_{i,K1}^h$ and $\mathbf{x}_{i,K2}^h$.

The reconstruction of the patch is done, then, according to the following steps.

1. Find the k-NN of $\mathbf{x}_i^l$, in the dictionary (external or internal). Let then $\mathbf{Y}_i^l$ be the matrix with these neighbors put as columns, and $\mathbf{Y}_i^h$ the corresponding HR neighborhood matrix.
2. Solve the following optimization problem to find the vector of weights $\mathbf{w}_i$ (the second and third terms are meant to impose consistency with the closest patches in the key frames):

$$\begin{aligned}\mathbf{w}_i = \arg\min_{\mathbf{w}} \big\{ &\|\mathbf{x}_i^l - \mathbf{Y}_i^l\mathbf{w}\|^2 + \mu_1\|\mathbf{x}_{i,K1}^h - \mathbf{Y}_i^h\mathbf{w}\|^2 \\ &+ \mu_2\|\mathbf{x}_{i,K2}^h - \mathbf{Y}_i^h\mathbf{w}\|^2\big\} \quad \text{s.t.} \quad \mathbf{w} \geq 0 \; . \end{aligned} \tag{4.1}$$

3. Reconstruct the HR patch:

$$\mathbf{x}_i^h = \mathbf{Y}_i^h\mathbf{w}_i \; .$$

The weight computation method resulting from (4.1) corresponds to the solution of a nonnegative least squares (NNLS) problem, as the NNLS problem presented in Section 2.2 when proposing the *NoNNE* SR algorithm (see Equation (2.7)). Here, however, we have three terms involved in the minimization: besides the usual LR approximation term, we have in fact two "temporal consistency terms" that impose the weight vector $\mathbf{w}$ to lead to a reconstructed patch similar to the HR patches found in the key frames.

According to the values of μ_1 and μ_2, more or less importance is given to the temporal consistency terms. Fixed a value of $\mu_{tot} = \mu_1 + \mu_2$, we decide to make μ_1 and μ_2 inversely proportional to the Euclidean distances w.r.t. the related LR patches:

$$\begin{aligned}\mu_1 &= \mu_{tot}\frac{\frac{1}{\|\mathbf{x}_{i,K1}^l - \mathbf{x}_i^l\|^2}}{\frac{1}{\|\mathbf{x}_{i,K1}^l - \mathbf{x}_i^l\|^2} + \frac{1}{\|\mathbf{x}_{i,K2}^l - \mathbf{x}_i^l\|^2}} \quad . \\ \mu_2 &= \mu_{tot} - \mu_1 \end{aligned} \tag{4.2}$$

We call this procedure motion estimation aided upscaling ("MEA" upscaling). The MEA method can be applied, as said, in both scenarios. In next sections, we assess its performance w.r.t. simple single-image upscaling, where each frame is upscaled independently without taking into account the other reconstructed frames.

4.2 Upscaling of a video sequence with HR key frames

In this section we deal with the problem of upscaling a video sequence in the case of *Scenario A*. In this scenario, as we can also see in Figure 4.1, we have a sequence of LR frames interlaced with periodic HR frames, appearing according to a certain frame frequency f_K. In Section 4.2.1, we compare different procedures implemented to this end, by presenting the *PSNR* values of the super-resolved frames averaged over several frames and the average processing. Scenario A, by its nature, lends itself to be analyzed in the compression context. Since we have periodic HR frames available, in fact, we can think that the rest of the frames have been intentionally down-scaled to low resolution in order to reduce the amount of data to encode. All frames are then brought back to high resolution on the decoder side, by upscaling the LR frames with a desired SR algorithm. We call this approach to video coding "SR approach", and compare, from a rate-distortion point of view, this scheme to the common case where the HR video source is directly encoded. This analysis is provided in Section 4.2.2, where the SR procedure adopted in the SR approach is the method showing the best performance in Section 4.2.1.

4.2.1 Comparison between different procedures

In order to upscale a video sequence with periodic HR key frames, we decide to adopt the neighbor embedding (NE) based SR procedure with nonnegative embedding and centered features described in Section 2.2 (the *NoNNE* algorithm), since it showed competitive performance and low-complexity characteristics.

Given this choice and an external dictionary, opportunely trained from a set of external images, we then consider three different procedures:

Proc. 1 NE-based SR with the external dictionary and the "usual" NNLS weights (i.e. nonnegative weights resulting from a least squares approximation of each LR patch),

Proc. 2 NE-based SR with the same external dictionary and "MEA weights" (still nonnegative weights, but computed with the motion-estimation-aided method described in Section 4.1), and

Proc. 3 NE-based SR with an *internal dictionary* built from the key frames and NNLS weights.

In all cases, the approach adopted is a single-image upscaling approach: each frame is separately super-resolved by means of a SR procedure, according to the principles presented in Section 1.1.2; to upscale N frames, then, N upscaling procedures are required. With Proc. 2, though, as said, the fact that the frames belong to a common video sequence is taken into account, by enforcing similar reconstructions when computing the weights of each patch reconstruction. In Proc. 3, instead, the video case is implicitly considered and exploited, since an internal dictionary of self-examples is built from the key frames, i.e. "real" HR patches coming from the video sequence itself are combined together to upscale a given patch in an intermediate frame.

The above-mentioned procedures have been tested with several sequences in CIF (Common intermediate format) format (352×288 pixels): the original sequence is downsized by a factor of 2 in each direction, i.e. to QCIF (Quarter CIF) format, except for the key frames, which stay in CIF format. The goal of the video SR procedure is to upscale the intermediate frames too to the CIF resolution. Table 4.1 summarizes the results of the simulations, where the columns 'Ext", "MEA", and "Int", refer to the procedures illustrated, in the order in which they have been mentioned. We consider a period of 16 frames (i.e. $f_K = \frac{1}{16}$), where Frame 1 and Frame 17 represent our key frames and the frames in between are those ones that have to be upscaled. As performance metrics, we reported the *PSNR* values of each super-resolved frame, the average *PSNR* of the whole group of pictures (GOP), and the average time per frame. The times given are for guidance only: they do not represent a particular achievement, since they are based on simulations conducted in Matlab with a non-optimized code, but they can anyway give an indication of the complexity of each procedure.

For the motion estimation step in the MEA procedure, i.e. to find the closest patches in the key frames, we adopted the Adaptive Rood Pattern Search (ARPS) algorithm of [87]. The ARPS algorithm is a fast block-matching algorithm, where, given an input image and a reference image, for each overlapping patch in the input image a motion vector (MV) pointing to the closest patch in the reference image is found. It consists of two sequential search stages:

1. an initial search, where we get a first estimate of the MV as the result of a search performed on macro-blocks, and
2. a refined local search, where a unit-size rood pattern is exploited repeatedly and unrestrictedly, until the final MV is found.

From the results in Table 4.1, we can observe a clear progression from Proc. 1 to Proc. 3. With respect to the usual NE-based procedure with external dictionary and weights computed to approximate the LR input patches (Proc.

	AKIYO			CITY			CONTAINER		
Frame	**Ext**	**MEA**	**Int**	**Ext**	**MEA**	**Int**	**Ext**	**MEA**	**Int**
1	-	-	-	-	-	-	-	-	-
2	37.60	40.83	51.45	30.10	30.86	32.01	28.44	32.18	40.84
3	37.64	40.73	51.11	30.10	31.06	32.79	28.44	31.85	36.94
4	37.62	40.73	50.67	30.14	30.96	32.04	28.47	31.57	35.36
5	37.72	40.80	49.26	30.14	31.10	32.92	28.45	31.47	34.98
6	37.78	40.77	49.11	29.97	30.81	32.43	28.44	31.71	35.63
7	37.73	40.75	48.83	29.90	30.57	32.20	28.44	31.91	36.90
8	37.73	40.75	48.51	29.82	30.60	32.23	28.42	31.98	37.88
9	37.68	40.67	48.02	29.77	30.65	32.46	28.45	31.94	38.10
10	37.58	40.43	47.21	29.79	30.79	32.55	28.41	31.95	37.85
11	37.52	40.34	46.93	29.94	30.71	32.23	28.41	31.83	36.52
12	37.57	40.39	46.91	29.76	30.80	32.71	28.42	31.59	35.33
13	37.45	40.21	46.64	30.29	31.09	32.44	28.44	31.49	34.89
14	37.47	40.28	47.15	30.37	31.32	32.88	28.45	31.65	35.58
15	37.41	40.18	46.48	30.60	32.02	33.94	28.43	31.94	37.39
16	37.32	39.94	45.68	30.32	31.20	32.16	28.38	32.09	40.98
17	-	-	-	-	-	-	-	-	-
Avg *PSNR*	37.59	40.52	48.26	30.07	30.97	32.53	28.43	31.81	37.01
Avg Time	16.3	24.3	40.0	16.8	28.7	44.6	14.2	24.0	36.4

Table 4.1: *PSNR* and time per frame for the video upscaling procedures considered in *Scenario A* - In the three procedures considered, the *NoNNE* algorithm is employed with, respectively, an external dictionary, the same external dictionary with "MEA weights", and internal dictionary derived from the key frames (i.e. frames 1 and 17).

1), the use of the key frames in the weight computation step, as done in the MEA method employed by Proc. 2, brings a consistent improvement (from about 1 to 3dB in average). In Proc. 2, though, we still combine patches coming from an external dictionary. When using actual patches coming from the video sequence itself, which is the case with the internal dictionary built in Proc. 3, we have even better results (a considerable gain, from 1.5 to even 7.7 dB, w.r.t. Proc. 2). As expected, we have the highest gain for those sequences like "Akiyo", where there is not much motion and patches in close frames are quite preserved. The advantage in using an internal dictionary can be observed also in Figure 4.3, where the performance of equally-sized (external and internal) dictionaries are reported, in terms of average reconstruction errors for single patch reconstructions, as the number of neighbors K of the NE procedure varies. The average error in the case of internal dictionary is considerably lower; in both cases, however, the error monotonically decreases with K.

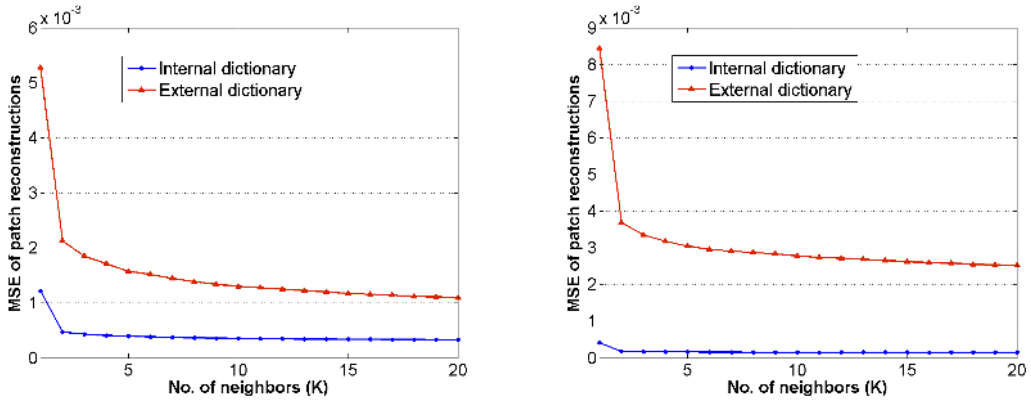


Figure 4.3: Average patch reconstruction error for a video frame upscaled by using an external and an internal dictionary - The neighbor embedding (NE) procedure is considered. The reconstruction error is clearly much lower in the case of internal dictionary; moreover, in this case, NE turns out to lead to even sparser representations.

As for the time performance (see again Table 4.1), the results are opposite. As expect, Proc. 1 requires the least amount of time, as "simple" single-image upscalings are performed by using an external pre-built dictionary. In the case of Proc. 2, where the MEA procedure is adopted, the running time of the algorithm increases by about 70%, due to the double weight compensation step: the goal of this step is to compute, for each frame to super-resolve, a double set of MVs, each one referring to one of the two key frames. In the case of Proc. 3, the average running time is even higher (about 250% of the time required by the external-dictionary procedure). Here, the additional complexity is due to the construction of the internal dictionary, an operation that has to be performed "online" for each GOP.

As an example, Figure 4.4 reports the visual results for a frame of the "Container" video sequence (Frame 8), by showing the same frame super-resolved with, respectively, Proc. 1, Proc. 2, and Proc. 3.

The images reported in Figure 4.4 confirm the quantitative results of Table 4.1. With respect to the frame super-resolved with Proc. 1 and Proc. 2, the result of the internal-dictionary procedure (Proc. 3) presents finer high-frequency details and sharper edges (e.g. see the motorboat in the upper left part of the image).

4.2.2 Analysis in the coding context

From the comparisons performed in Section 4.2.1, it is clear that, in the case of Scenario A (the LR video frames are interlaced with periodic HR frames), using an internal dictionary is particularly favorable. For a given LR frame to

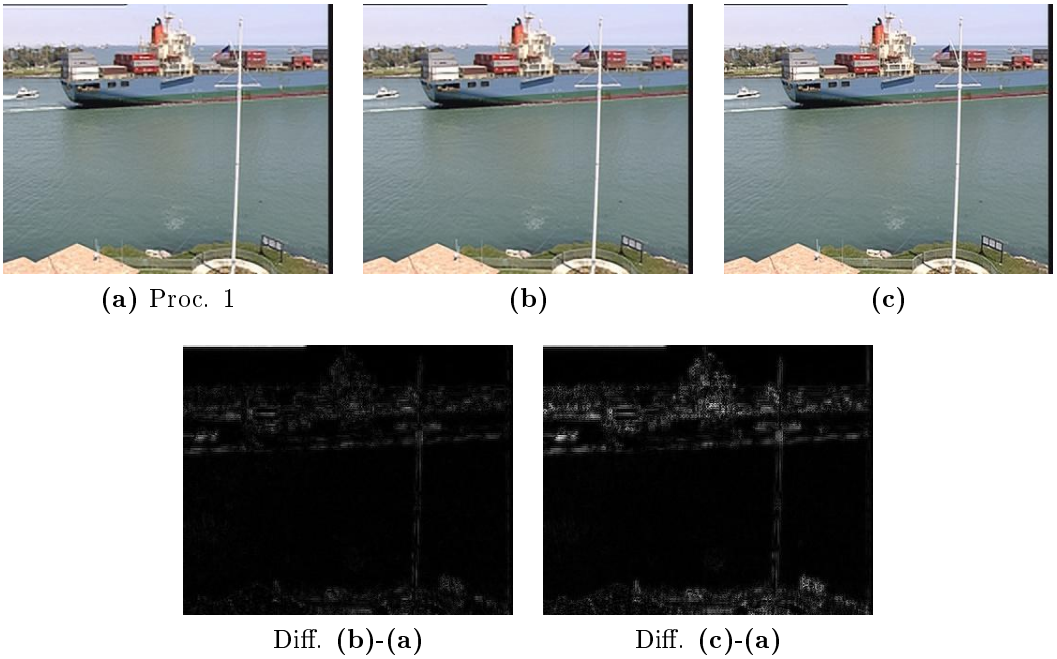


Figure 4.4: Visual results for a frame of the "Container" video sequence, upscaled within Scenario A. - The super-resolved images for each of the 3 procedures considered are reported in the top row; the two images in the bottom row point out the mutual differences.

be upscaled, the internal dictionary is built from the two neighbor key frames, by sampling pairs of patches from the key frames themselves and down-sampled versions of them.

Among the procedures presented, Proc. 3 is the one that makes use of an internal dictionary. The method is close to the SR algorithm of Hung *et al.* proposed in [86]. In [86], for a given patch, the search for the closest patches is performed by restricting it to windows positioned on the key frames (the central positions are given by computed motion vectors): the two best matching patches are selected, one for each key frames, and combined according to distance-depending weights. Differently than [86], we instead proposed with Proc. 3 to perform the neighbor search globally, by considering the dictionary formed by the two current key frames as a whole. This solution leads to a simpler implementation and a complexity decrease. Indeed, no motion estimation is required and since the dictionary is unique for all the patches of an intermediate frame, we can compute the neighbors all at once with fast neighbor search algorithms. Moreover, we believe that, although no motion estimation is taken into account, the temporal consistency between frames is nevertheless respected. The best matches overall are reasonably also those ones we would choose by first selecting a search area by

motion estimation.

Video SR in Scenario A is particularly suitable for being studied in a coding context. As some HR frames are accessible by the end-user, in fact, it is reasonable to think that the HR source is "somewhere" available. In this case, it means that the video scenario has been intentionally created, since we want to adopt a "SR approach" to the transmission of a HR video sequence. The approach consists in:

1. Down-sampling large part of the HR sequence, by keeping only a few HR key frames,
2. Encoding and transmitting this "mixed" sequence (with a lower coding cost), and
3. Subsequently applying SR to retrieve the original sequence.

We then want to evaluate if the *SR approach*, where Proc. 3 is used as upscaling method, is a convenient solution, rather than directly encoding the HR sequence. In other words: can SR be a useful tool in video compression?

For making this analysis, we use the innovative High Efficiency Video Coding (HEVC) video compression standard [88]. We consider two configurations of HEVC:

- The efficient HEVC *random-access profile*, with inter coding enabled, and
- The HEVC *all-intra configuration*, with all frames coded independently in intra mode, corresponding to some particular application profiles with a low-delay/low-complexity requirement (e.g. digital cinema).

In our tests we considered HR sequences in CIF format (352×288). The group of pictures (GOP), as well as the intra-period in the codec, is composed of 32 frames (2 key frames that serve as reference frames and 30 intermediate frames). We evaluated then the following two cases, in terms of rate-distorsion (RD) performance of the reconstructed sequences.

- *Direct HR encoding*: the CIF sequence is directly encoded and transmitted.
- *SR approach*: the CIF sequence is down-sampled to the QCIF format (176×144); the QCIF sequence is encoded and transmitted, as well as some CIF intra-coded key frames; the proposed video SR algorithm is then applied on the decoded frames to re-upscale to CIF format.

In the first case, different reconstruction qualities are achieved, when varying the quantization parameter (QP) of the encoded CIF sequence, so obtaining a single RD curve. In the SR approach, instead, we have two parameters that we can play with: the QP of the intra-coded CIF key frames and the QP of the encoded QCIF sequence. By fixing, from time to time, the quality of the intra-coded key frames, we can draw a set of curves.

Figure 4.5 shows the RD curve for the HR encoded case (in black) and the set of RD curves for the SR approach, for the *Hall* video sequence, in the inter coding

(random-access) configuration. For each of the RD curves of the SR approach, we can choose the best "operating point", so identifying an optimal pair of QP (QP of the CIF frames and QP of the QCIF sequence). The corresponding values of bit-rate (in kbps) and PSNR for the HR encoded sequence and the four operating points of the SR approach are reported in Table 4.2.

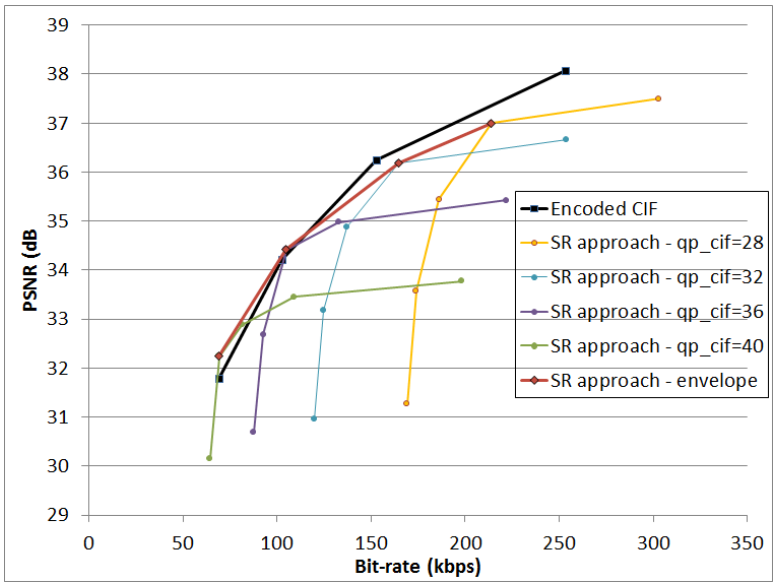


Figure 4.5: RD comparison between direct encoding and SR approach (*Scenario A*) for the *Hall* sequence, HEVC employed in the random-access configuration - For the SR approach we have a set of curves, whose "envelope" gives similar results than directly encoding the HR sequence.

QP	Bit-rate	PSNR
28	253.2	38.08
32	152.8	36.25
36	102.8	34.21
40	69.3	31.80

(A)

QP CIF	QP QCIF	Bit-rate	PSNR
28	24	214.0	37.00
32	24	164.7	36.18
36	28	104.9	34.42
40	32	69.3	32.24

(B)

Table 4.2: Bit-rate (kpbs) and *PSNR* (dB) for different QP values in the video coding analysis of Scenario A - The cases considered are: HR direct encoding (A); for the four operating points resulting in the SR approach (B).

All the chosen operating points together give an "envelope" curve that summarizes the performance of the SR approach, which we can compare to the black curve of the HR encoding approach. As we can see, the SR approach gives a slight

improvement for low bit-rates, when, while having the same poor encoding, SR can actually helps improving the reconstruction quality.

Figure 4.6 reports similar curves, the black RD curve for the HR encoding case and the RD envelope for the SR approach, with the all-intra configuration, still for the *Hall* sequence. Here, since the encoding is not fully efficient, we can actually save much bit-rate with sending the "mixed" QCIF/CIF sequence and letting SR do part of the job at the decoder. In this case, the SR approach shows better RD performance also for mid-range bit-rate values.

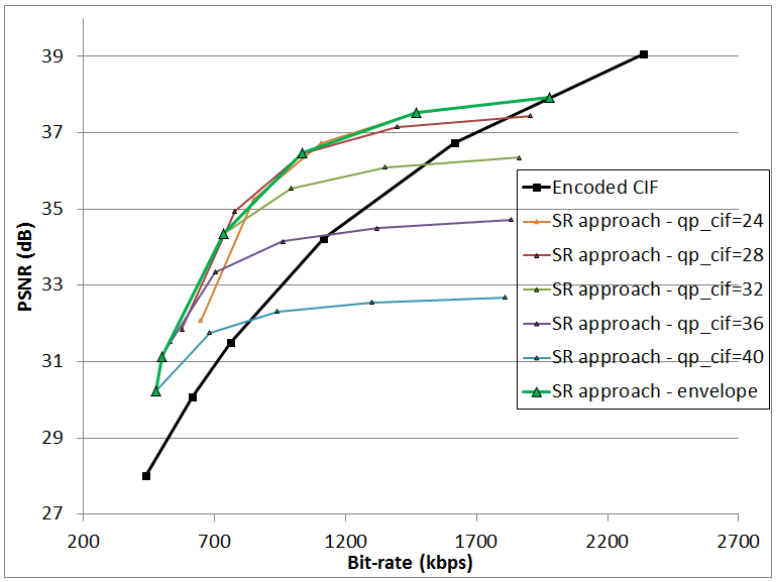
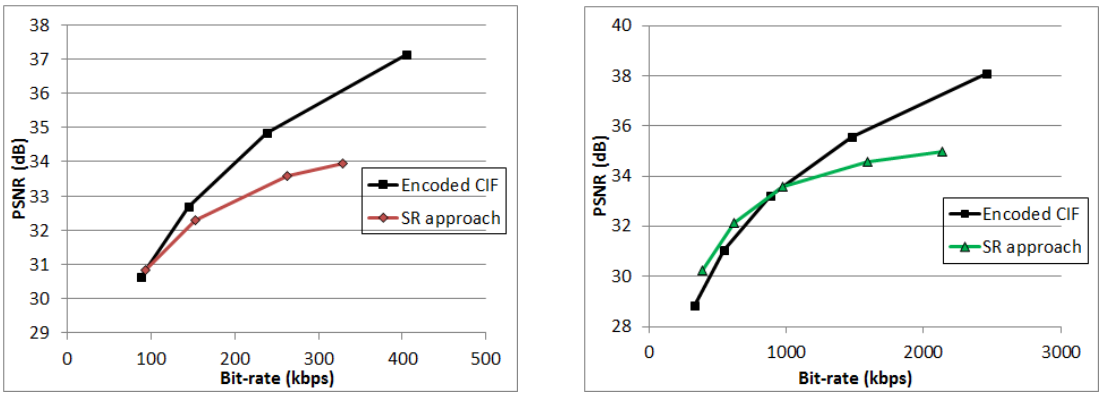


Figure 4.6: RD comparison between direct encoding and SR approach (*Scenario A*) for the *Hall* sequence, HEVC employed in the the all-intra configuration - The SR approach shows here a certain gain for mid-range *PSNR* values.

Figure 4.7 shows the results in terms of RD curves for the *Foreman* sequence, in the two configurations considered (inter coding and all-intra). For this sequence, the performance of the SR approach is worse. We can observe a certain gain for low bit-rates in the all-intra configuration, but the performance is clearly lower than the HR encoding case in the random-access configuration. Apparently, the *Foreman* sequence is a sequence which is more difficult to super-resolve (the HR details are difficult to retrieve), so the SR benefit cannot compensate the loss of quality due to the down-sampling process, while the saving of bit-rate being not so relevant.

Finally, Figure 4.8 reports as an example the visual results for two frames of the *Hall* sequence, one for each configuration considered: random-access inter (left) and all-intra configuration (right). In both cases, the bit-rates achieved



(inter coding enabled) (all-intra)

Figure 4.7: RD comparison between direct encoding and SR approach (*Scenario A*) for the *Foreman* sequence, HEVC used with two different configurations - For the random-access configuration (left) the SR approach shows worse performance; things get slightly better for the all-intra configuration (right).

(the supposed qualities) for the case of HR direct encoding via HEVC and the SR approach are comparable, or slightly smaller for the SR approach. As we can observe from the images, employing SR on down-scaled frames turns out to produce generally more blurred images. However, the SR approach does not present some compression artifacts, which are instead visible in the case of direct encoding (see the baseboard on the bottom-left corner). Also when playing the whole video sequence, the results of the SR approach are acceptable. As a matter of fact, we did not observe any particular flickering problems, thus meaning that using an internal dictionary built from key frames (and so "real patches" of the sequence) is already a way to automatically impose a sort of temporal consistency.

The simulation conducted gave some insights, in order to answer the initial question of this section, i.e. whether super-resolution can be a useful tool in video compression. When comparing the SR approach and the direct encoding scheme in terms of rate-distortion performance, the former presents, in fact, a certain gain (for low or mid-range bit-rates) in the *all-intra configuration*, i.e. when all frames are encoded in intra mode. When adding also inter coding (*random-access profile* of HEVC), however, this *PSNR* gains tends to disappear. In terms of visual perception, the frames decoded and subsequently upscaled via SR generally look smoother and contain less artifacts. Moreover, the SR approach can be helpful in reducing the flickering of the video.

Original CIF

Original CIF

HECV encoding (inter, QP=40)

HECV encoding (intra, QP=36)

SR approach (comparable bit-rate)

SR approach (comparable bit-rate)

Figure 4.8: Visual comparison between direct HR encoding and SR approach - Two frames of the *Hall* sequence are considered: frame no. 24 (left) and frame no. 32 (right).

4.3 Upscaling of a video sequence with only LR frames

In this section we deal with the problem of upscaling a video sequence in the case of *Scenario B* (Figure 4.2), i.e. when we have a set of LR frame to upscale and we cannot take advantage of strategic HR frames, as in *Scenario A*.

Here too, we adopt a "single-image strategy": to upscale a video sequence, no further modifications are done to the single-image SR algorithms that we described in the previous chapters, but the same algorithms are applied, frame by frame, to enlarge the resolution of the whole sequence. As pointed out in [56, 45], in fact, we also observed that, when an output video is stable at its low-frequency component, an overall stability is achieved and no flickering effect occurs.

For the video upscaling problem, we then consider three different single-image SR methods:

- Our *NoNNE* algorithm presented in Section 2.2,
- *NoNNE* + the MEA weight computation procedure described in 4.1, and
- Our "double-pyramid" internal-dictionary algorithm presented in Chapter 3.1.

Table 4.3 summarizes the results of the video uspcalings, where the methods above-mentioned have been employed to super-resolve several QCIF test sequences to the CIF format (enlargement by a factor of 2 in both directions). As performance metrics, we reported again the *PSNR* values of each super-resolved frame, the average *PSNR* of the whole GOP, and the average time per frame, as an approximate estimate of the complexity of each method. The columns 'Ext", "MEA", and "Int", indicate the different SR methods used, respectively the NoNNE algorithm, NoNNE+MEA, and the double-pyramid algorithm. In the latter, an internal dictionary is built for each frame, according to the double-pyramid structure. The dictionary thus built is inevitably less "powerful" then the dictionary for the internal procedure in the case of Scenario A (Proc. 3. In this case, in fact, while building the internal dictionary, we do not have "real" HR patches available: the pairs of patches composing the dictionary are then less efficient in representing the mapping between the LR frames and the unknown HR frames.

As we can see from the results of Table 4.3, in the case of Scenario B we do not have a clear difference in terms of quality performance between the various procedures tested. With respect to the external-dictionary-based NoNNE algorithm, the double-pyramid algorithm, which uses an internal dictionary of self-examples and progressively reconstructs the output image in several passes, brings a small, almost negligible, improvement, in the order of 0.1/0.2 dB (also

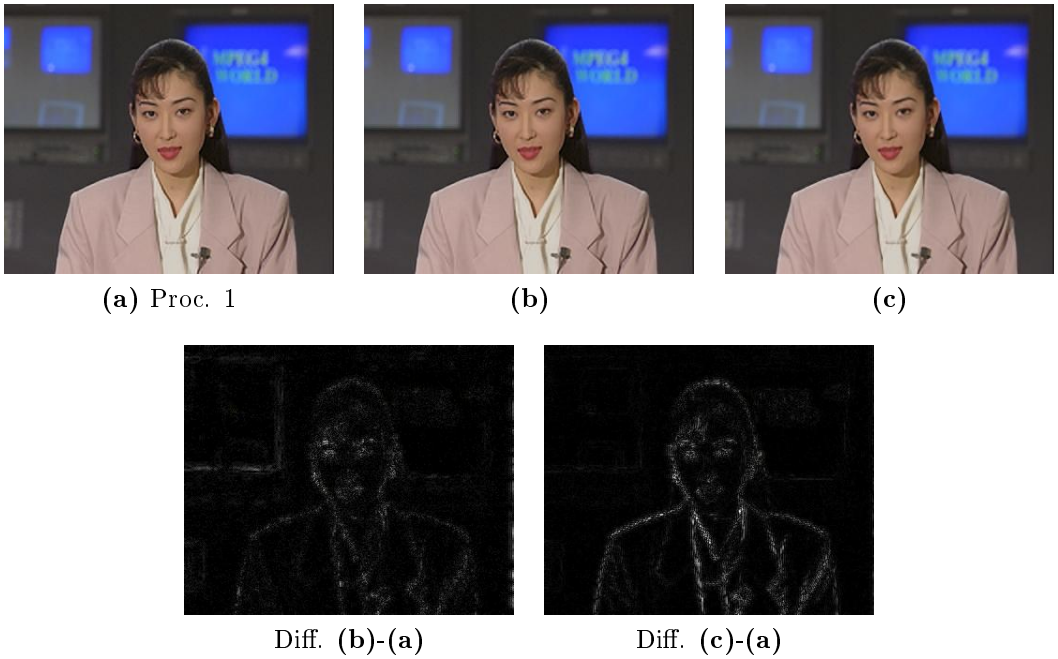
	AKIYO			CITY			CONTAINER		
Frame	**Ext**	**MEA**	**Int**	**Ext**	**MEA**	**Int**	**Ext**	**MEA**	**Int**
1	37.58	37.58	37.72	30.14	30.14	30.22	28.48	28.48	28.74
2	37.60	37.62	37.73	30.10	30.10	30.15	28.44	28.61	28.72
3	37.64	37.59	37.74	30.10	30.14	30.16	28.44	28.61	28.72
4	37.62	37.59	37.72	30.14	30.18	30.22	28.47	28.62	28.73
5	37.72	37.71	37.87	30.14	30.21	30.25	28.45	28.62	28.76
6	37.78	37.72	37.89	29.97	30.00	30.03	28.44	28.65	28.76
7	37.73	37.72	37.89	29.90	29.90	29.98	28.44	28.63	28.75
8	37.73	37.72	37.87	29.82	29.84	29.91	28.42	28.62	28.73
9	37.68	37.66	37.85	29.77	29.79	29.84	28.45	28.60	28.71
10	37.58	37.57	37.75	29.79	29.85	29.93	28.41	28.59	28.71
11	37.52	37.54	37.73	29.94	29.95	29.96	28.41	28.58	28.72
12	37.57	37.57	37.73	29.76	29.79	29.86	28.42	28.58	28.72
13	37.45	37.48	37.62	30.29	30.30	30.28	28.44	28.60	28.77
14	37.47	37.51	37.67	30.37	30.42	30.43	28.45	28.61	28.75
15	37.41	37.43	37.64	30.60	30.67	30.64	28.43	28.61	28.72
16	37.32	37.27	37.57	30.32	30.38	30.34	28.38	28.53	28.66
17	37.22	37.20	37.64	30.05	30.04	30.10	28.34	28.33	28.67
Avg *PSNR*	37.57	37.56	37.74	30.07	30.10	30.14	28.43	28.58	28.73
Avg Time	18.5	23.7	96.5	19.0	31.6	199.2	19.1	32.1	157.5

Table 4.3: *PSNR* and time per frame for the video upscaling procedures considered in *Scenario B* - The three procedures considered are, respectively, the NoNNE algorithm, NoNNE+MEA, and the double-pyramid algorithm.

from the results of Table 2.4, in fact, we could observe that the advantage of using an internal dictionary tends to be less evident for small scale factors). Also the motion-estimation-aided weight computation method does not bring anything to the NE approach. This result was somehow predictable, since with the MEA method we rely upon previous reconstructions, which in this case, differently than the in Scenario A, are simply obtained with the same SR algorithm and do not contain any more "certain" information. By also considering the time complexity issue, then, the "simple" NoNNE algorithm appears to be the most reasonable solution in Scenario B, as it presents similar results while having a considerably lower complexity w.r.t the other two procedures.

Some visual results, related to a frame of the "Akiyo" sequence (Frame 11) super-resolved with the different methods, are reported in Figure 4.9.

The images showed in Figure 4.9 confirm that there is a substantial equality between the three methods analyzed. From the second differential image (in the bottom-right part of Figure 4.9), which shows the differences between the



(a) Proc. 1 **(b)** **(c)**

Diff. **(b)-(a)** Diff. **(c)-(a)**

Figure 4.9: Visual results for a frame of the "Akiyo" video sequence, upscaled within Scenario B. - The super-resolved images for each of the 3 procedures considered are reported in the top row; the two images in the bottom row point out the mutual differences.

NoNNE result and the result obtained with the double-pyramid algorithm, we can however see that the internal-dictionary procedure is able to reconstruct sharper edges (most of the differences lies along the contours of the woman), that justifies the 0.2-dB difference observable in Table 4.3.

As in Section 4.2.2 we want to analyze video SR of sequences belonging to Scenario B in a compression context. Here too, given an original HR sequence in CIF format, we consider two possible approaches:

- A *direct encoding* approach, where the CIF sequence is directly encoded and transmitted, and
- A *SR approach*, where the CIF sequence is down-sampled to the QCIF format, and the built QCIF sequence is encoded, transmitted, and re-upscaled via SR on the decoder side.

In the SR approach, the NoNNE algorithm, which offers slightly worse performance but has been proved to be more favorable to low-complexity purposes, is used.

For each of the approaches, we have a related parameter that we can tune in the encoding phase, thus varying accordingly the video quality and the coding cost: the quantization parameter (QP) of the video coder. In particular, in

the direct encoding approach we can vary the QP when encoding the original CIF sequence, while for the SR approach we can act on the QP of the QCIF sequence. By computing the coding cost and the average *PSNR* of the decoded video sequence (decoded and super-resolved in the case of the SR approach), we obtain the values reported in Table 4.4. For this test we considered the best case that we observed in the case of Scenario A, i.e. the use of the all-intra configuration, where all frames are encoded as intra frames, and the "Hall" video sequence, which have prove to be an easy sequence to super-resolve.

QP	**Bit-rate**	**PSNR**
28	2337.2	39.06
32	1614.6	36.74
36	1116.7	34.23
40	760.7	31.50
42	615.5	30.08
45	436.6	28.00

(A)

QP	**Bit-rate**	**PSNR**
20	1824.7	31.13
24	1294.1	30.83
28	924.6	30.16
32	654.2	29.41
36	447.8	28.02
40	257.7	26.08

(B)

Table 4.4: Bit-rate (kpbs) and *PSNR* (dB) for different QP values in the video coding analysis of Scenario B - The cases considered are: direct encoding of the CIF sequence (A), and SR approach (B).

The values in 4.3, reported visually, originates the plot of Figure 4.10, where we have a rate-distortion comparison between the direct encoding approach and the SR approach.

From what we can observe from Figure 4.10 and the values of Table 4.4, the R-D comparison is totally in favor of the direct encoding approach, although using the all-intra configuration of HEVC and thus a not extremely efficient coder. In this case, with only LR frames available, the SR algorithm does not appear to be able to achieve a satisfactory image quality.

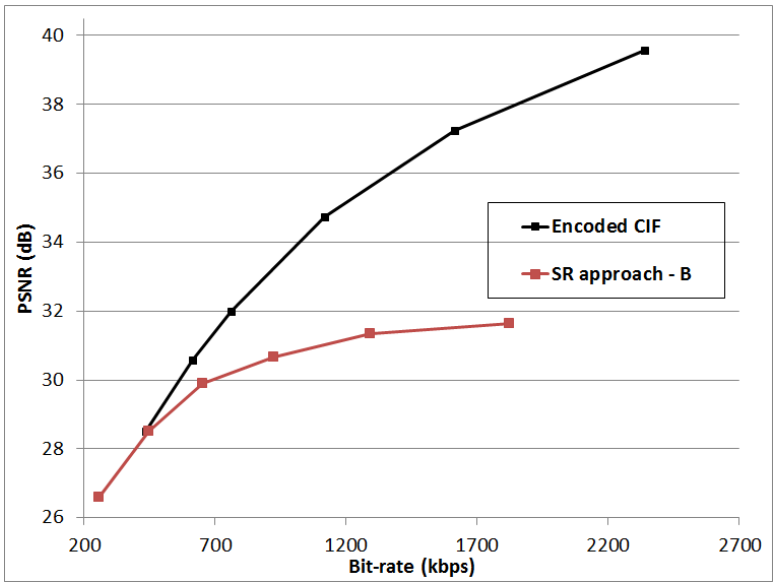


Figure 4.10: RD comparison between direct encoding and SR approach (*Scenario B*) for the *Hall* sequence, HEVC employed in the the all-intra configuration - The SR algorithm, applied to a sequence of only LR frames, does not allow to reach satisfactory *PSNR* values.

4.4 Conclusion

In this chapter we dealt with the application to the video case of the concepts and the algorithms related to single-image SR, developed in the previous chapters of this manuscript. In particular, we made use of the NoNNE algorithm presented in Section 2.2, based on an external dictionary and a nonnegative neighbor embedding (NE) procedure, and the "double-pyramid" algorithm described in Section 3.1, based on an internal dictionary. In addition to these methods, we also proposed in Section 4.1 a weight computation method to be used in NE-based procedures, which also takes into account a sort of temporal consistency criterion, by guiding each local reconstruction. We called this method motion-estimation-aided (MEA) procedure, since close patches are found in the key frames via motion estimation, in order them to play as references for a certain patch to super-resolve.

To upscale a video sequence, we then adopted a simple strategy, where a single-image SR procedure is repeated several times in order to obtain, for each frame of the sequence, its HR version. By adopting this strategy, although not optimized in terms of complexity (the similarity of neighbor frames is not exploited in order to possibly upscale a group of frames at once), the fact that for all the frames we adopt the same SR method (and possibly the same dictionary

of patch correspondences) makes the reconstructed frames consistent between each other, and a global stability of the output video sequence, at least in terms of low frequencies, is achieved. The same considerations about the adoption of single-image SR procedures for video upscaling are also made in [56, 45].

We considered two upscaling scenarios:

- A scenario where a sequence of LR frames also contains periodic HR "key frames" (**Scenario A**), and
- A scenario where we have a video sequence of only LR frames (*Scenario B*).

.

For Scenario A we compared two methods, both based on nonnegative NE, using alternatively an external dictionary (in this case we have then the "pure" NoNNE algorithm) and an internal dictionary, built from the available HR key frames. For the external-dictionary method we also took into account to have in addition the MEA procedure. In this comparison the internal-dictionary algorithm turned out to be the clear winner. If we have some HR frames of the sequence available, it is in fact totally reasonable to use them, to build a dictionary of patches upon. For Scenario B we instead compared the NoNNE algorithm, with or without the MEA procedure as a feature, and the double-pyramid algorithm, that constructs an internal dictionary of self-examples for each key frames. In this case, the three procedures turned out to give quite similar quality results; the "simple" NoNNE, then, has to be preferred to the other algorithms for its lower complexity. As for the internal-dictionary algorithm, instead, a more "sophisticated" way to build the dictionary needs probably to be designed, by merging the pyramids of several frames into a single dictionary that keeps the most meaningful self-examples.

For the two scenarios considered, we also performed an analysis in the coding context, i.e. when the video sequences are encoded for compression purposes. We then compared the case where an original HR sequence is directly encoded, with the efficient and advanced HEVC codec, to the case where the video sequence, before being encoded, is down-scaled (by possibly keeping some HR frames, as in Scenario A). On the receiver side, after decoding the sequence, SR is then employed to bring the whole sequence back to the original HR format. We called this second scheme the "SR approach". The coding cost being equal, with the SR approach, since we have a smaller amount of data to encode, we can afford a better encoding quality (i.e. a smaller quantization parameter). The question, then, is: is it better to super-resolve a good-quality LR sequence or to directly have the original HR sequence, although with average quality? While the task of super-resolving a video sequence of only LR frames (Scenario B) turned out to be difficult, interesting results came in the case of Scenario A, especially when HEVC is employed in the all-intra configuration, that still corresponds to some

application profiles. In this case, the SR approach outperformed for low and mid-range bit-rates the direct encoding approach, thus proving that under certain conditions SR can be effectively used as a compression tool.

Chapter 5

Classification of the learning techniques used in the proposed SR methods

In this last chapter we classify some techniques belonging to the machine learning area, which we studied and used while developing algorithms for super-resolution purposes. In particular, after introducing in Section 5.1 some concepts about the learning paradigm, in Section 5.2 we present regression as a method to perform *supervised learning*, by relating it to the learning-based SR algorithms described in Section 1.2.2.1. Later, in Section 5.3 we provide a compendium of a few *unsupervised learning* techniques that we particularly studied, notably some methods to perform dictionary learning and clustering. Within the framework of these techniques, we finally present in Section 5.3.5 a new method designed by us, to learn a dictionary of nonnegative atoms particularly suitable for sparse representations. The new method, that we called *K-WEB* (as it involves the computation of K weighted barycenters), is assessed by performing a comparison with other state-of-the-art methods in Section 5.3.6.

5.1 Learning from data

Modern science and engineering are based on *first-principle* models to describe physical, biological, and social systems. Such an approach starts with a basic scientific model (e.g. Newton's laws of mechanics), and then builds upon it various applications. Under this approach, experimental data (measurements) are used to verify the underlying first-principle models and to estimate some of the model parameters that are difficult to measure directly. However, in many applications the underlying first principles are unknown, or the systems under study are too complex to be mathematically described. To overcome this problem, in

the recent decades a new paradigm has emerged: *learning from data*. Nowadays, large amounts of data can be easily collected and processed, and we can use these readily available data to derive models, by estimating useful relationships between input and output variables. We are then witnessing a gradual shift in scientific methodology, from the classical modeling based on first principles (and subsequent validation with data) to directly deriving models from data.

Machine learning is a scientific discipline that concerns the construction and study of systems that can "learn from data", i.e. the design of algorithms that can extract information automatically. The learning process usually consists of two stages: a learning/estimation stage, where a model is learned from training samples, and a prediction stage, where predictions are made for future or test samples on the base of the model learned. As for the learning stage, we can basically distinguish three methodologies for learning empirical models from data.

- *Statistical model estimation*, which aims at developing adaptive learning algorithms, by extending the classical statistical and function approximation framework,
- *Predictive learning*, whose approach, originally developed by practitioners in the field of neural networks in the late 1980s, focuses on estimating models with good generalization capability (rather than "strictly true" models as in the case of statistical model estimation), and
- *Data mining*, which, in its narrowest meaning, attempts to extract a subset of data samples (from a give a large data set) with useful or interesting properties.

In this chapter we present some predictive learning techniques that we had the chance to study during this doctorate. The example-based approach to super-resolution described in Section 1.2 can be related to prediction learning too. For each LR patch in the input image to super-resolve, in fact, we want to find, by exploiting the training samples of the dictionary, an estimating model which is general enough to predict the corresponding HR output patch.

In predictive learning, there are two common types of problems :

- Supervised learning, and
- Unsupervised learning.

Supervised learning is used to estimate an unknown (input, output) mapping from known (input, output) samples. Classification and regression tasks fall into this group. The term "supervised" denotes the fact that output values for training samples are known (i.e., provided by a "teacher" or a system being modeled). Under the *unsupervised learning* scheme, instead, only input samples are given to a learning system, and there is no notion of the output during learning. The goal of unsupervised learning may be to approximate the probability distribution

of the inputs or to discover natural structures (i.e., clusters) in the input data. In Section 5.2 we debate regression as a broad family of supervised learning techniques, which deal with real-valued outputs (as opposed to categorical output variable, as in the case of classification).

5.2 Regression methods

In predictive learning, we have a different naming convention for the prediction task, according to the output type. Broadly speaking, *regression* is when we predict quantitative outputs, and *classification* when we predict qualitative outputs. An input variable, or *predictor*, is typically denoted by the letter X; if X is a vector of variables, the notation with a subscript X_i indicates one of its components. Quantitative outputs, or *responses*, are instead indicated by the letter Y. In the supervised learning scheme we assume that the inputs, which are measured or preset, have some influence on one or more outputs. We then choose a model to express this influence, and use a set of training data $(x_1, y_1) \dots (x_N, y_N)$ (observed values are written in lowercase) to estimate the parameters of this model. Once the model is "trained", we can use it to predict the response of new inputs.

The problem expressed in this way is closely related to the problem of the single patch reconstructions in example-based SR. In the latter case, the new input-output pairs are represented by a LR test patch in the LR input image and the corresponding HR patch that we want to predict, and the set of training data is given by the whole dictionary or, if a nearest neighbor search is performed, by coupled neighborhoods of LR and HR patches. In the next sections we review some basic regression methods, and relate them to the learning-based reconstruction methods we described in Section 1.2.2.1: neighbor embedding (NE) and direct mapping (DM).

5.2.1 Linear regression

In linear regression we assume that the model is linear in the parameters. Let first suppose that we have only one output variable Y and a vector of p inputs $X^{\mathrm{T}} = (X_1, X_2, \dots, X_p)$; the linear regression model has then the form

$$Y = f_0 + \sum_{i=1}^{p} X_i f_i \,, \tag{5.1}$$

where f_0 is the intercept, also known as *bias*, and $\{f_i\}_{i=1}^{p}$ is a set of p parameters (we then have $p+1$ parameters in total). The model (5.1) can also be expressed in vector form

$$Y = X^{\mathrm{T}}\mathbf{f} \,, \tag{5.2}$$

where, for the sake of simplicity, the parameter vector $\mathbf{f}$ is a $(p+1)$-dimensional vector including the intercept too, and X has been rewritten to include a constant as well, i.e. $X^{\mathrm{T}} = (1, X_1, \ldots, X_p)$.

To train the model, we fit it to the available set of N training data

$$(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_N, y_N)$$

(note that each pair of training data consists in this case of an input vector and a real-valued output), with the popular method of least squares, i.e. by minimizing the sum squared errors

$$\mathrm{RSS}(\mathbf{f}) = \sum_{i=1}^{N} (y_i - \mathbf{x}_i^{\mathrm{T}}\mathbf{f})^2 = (\mathbf{y} - \mathbf{X}\mathbf{f})^{\mathrm{T}}(\mathbf{y} - \mathbf{X}\mathbf{f}) \ , \tag{5.3}$$

where $\mathbf{X}$ is and $N \times (p+1)$ matrix with in each row an input vector, and $\mathbf{y}$ is the N-vector of the outputs in the training set.

The solution to (5.3) is easy to derive. If $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is nonsingular, we have:

$$\hat{\mathbf{f}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \ . \tag{5.4}$$

For a new arbitrary input vector $\mathbf{x}_0$, its predicted response can then be easily computed by using the parameter vector in (5.4), i.e. $\hat{y}(\mathbf{x}_0) = \mathbf{x}_0^{\mathrm{T}}\hat{\mathbf{f}}$.

If there are multiple outputs, i.e. $Y = [Y_1, Y_2, \ldots, Y_m]$, the above analysis stays valid, as it can be carried for each output separately. When we collect all the equations of the estimated parameters together, there holds

$$\begin{cases} \hat{\mathbf{f}}_1 = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}_1 \\ \quad \vdots \\ \hat{\mathbf{f}}_m = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}_m \end{cases} , \tag{5.5}$$

where $\hat{\mathbf{f}}_1, \ldots, \hat{\mathbf{f}}_m$ are m $(p+1)$-dimensional vectors of parameters, each one delegated to predict a single output variable, and $\mathbf{y}_1, \ldots, \mathbf{y}_m$ are m collections of N outputs taken from taken data, each one related to an output variable.

The set of formulas in 5.5 can simultaneously be rewritten in the following compact matrix form:

$$\hat{\mathbf{F}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y} \ , \tag{5.6}$$

where $\hat{\mathbf{F}}$ is a $(p+1) \times m$ parameter matrix and $\mathbf{Y}$ is a $N \times m$ response matrix collecting row by row N m-dimensional output vectors. Given a test input vector $\mathbf{x}_0$, the estimated response vector $\hat{\mathbf{y}}_0$ will then be in this case:

$$\hat{\mathbf{y}}_0^{\mathrm{T}} = \mathbf{x}_0^{\mathrm{T}}\hat{\mathbf{F}} \ . \tag{5.7}$$

The multivariate case of the linear regression model, whose solution is given in (5.6), takes the name of *multilinear regression* (MLR). The prediction equation written in (5.7) is totally similar to the SR patch reconstruction method that in Section 1.2.2.1 we called direct mapping (DM). In a DM reconstruction, given a LR input patch (in this case $\mathbf{x}_0$), we find its related HR output patch (in this case $\hat{\mathbf{y}}_0$, by applying to the former a direct function. In the double-pyramid algorithm described in Chapter 3, the mapping function chosen was, indeed, linear. Note, in fact, the the equation for the patch reconstructions of the double-pyramid algorithm (3.8) is exactly the same as (5.7), with the only difference that vectors and matrices are represented symmetrically .

5.2.2 k-nearest neighbor regression

Nearest-neighbor (NN) methods represent another approach to regression. Here too we suppose to have a training set $\mathcal{T} = \{(\mathbf{x}_1, \mathbf{y}_1) \dots (\mathbf{x}_N, \mathbf{y}_N)\}$, generally composed by pairs of input and output vectors of variables. Given a input test vector $\mathbf{x}_0$, the idea is to look for the k closest observations in the input space and use the corresponding known responses to form the output $\hat{\mathbf{y}}_0$. In the basic case, the output test vector is the result of a simple average:

$$\hat{\mathbf{y}}_0 = \frac{1}{k} \sum_{\mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}_0)} \mathbf{y}_i \ , \tag{5.8}$$

where $\mathcal{N}_k(\mathbf{x}_0)$ is the neighborhood of $\mathbf{x}_0$ defined by the k closest points $\mathbf{x}_i$ in the training set $\mathcal{T}$.

The philosophy of the k-NN regression is in the same spirit of the neighbor embedding (NE) approach we described in Section 1.2.2.1. An output vector is not directly derived by its related input vector, but the input vector is used only at a first stage (in this case to select only the a subset of observations in $\mathcal{T}$); then this knowledge in "transferred" to the output (HR) space and only the known responses are used to generate the final output vector.

In the NE approach we have been using, we do not have a simple average operation but also some weights appear: these weights are computed in the input (LR) space and "blindly" transferred to the output space. The equation of k-NN regression, in order to formally express the NE-based SR reconstruction method, then becomes:

$$\hat{\mathbf{y}}_0 = \sum_{\mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}_0)} w_i(\mathbf{x}_0, \mathbf{x}_i) \cdot \mathbf{y}_i \ , \tag{5.9}$$

where each known response $\mathbf{y}_i$ is now weighted by a personal coefficient that depends on the corresponding input vector in the training set $\mathbf{x}_i$ and the input test vector $\mathbf{x}_0$.

The equation (5.9) corresponds in fact to the equation (2.3) that we gave in Section 2.1, when presenting the general NE procedure for SR. In our designed nonnegative NE method, described in Section 2.2.2, the weights related to the selected observations (the candidates found in the dictionary via NN search) are computed all the once a single weight vector, by solving the following nonnegative least squares problem:

$$\mathbf{w}_{\mathbf{x}_0} = \arg\min_{\mathbf{w}} \|\mathbf{x}_0 - \mathbf{X}_{\text{set}}\mathbf{w}\|^2 \quad \text{s.t.} \quad \mathbf{w} \geq 0 \;, \tag{5.10}$$

where $\mathbf{X}_{\text{set}}$ is a matrix containing the k neighbors of $\mathbf{x}_0$ selected from the training set.

5.3 Some methods for unsupervised learning

In this section we present some *unsupervised learning* techniques. Unlike supervised learning, here there is no notion of the relation between inputs and outputs, but we only have a data matrix $\mathbf{Y}$ (the different signals are usually arranged as columns), from which we try to extract some meaningful information. Clustering and dictionary learning (that we can see as the extraction of relevant "atoms" from the available data) are typical unsupervised learning problems.

In sections 5.3.1, 5.3.2, and 5.3.3, we quickly revise three well-known learning algorithms, respectively, K-means clustering, *K-SVD*, and nonnegative matrix factorization (NMF). NMF is then considered with a supplementary sparsity constraint in Section 5.3.4. In Section 5.3.5 we finally present a novel algorithm, as an alternative to the methods described in Section 5.3.4, to learn a dictionary of nonnegative atoms suitable for sparse representations; experimental comparisons are provided in Section 5.3.6.

5.3.1 *K*-means clustering

Originally conceived as a method of *vector quantization* (VQ), K-means clustering is a popular technique for cluster analysis. The aim of K-means clustering aims is to partition N observations ($\mathbf{Y} = \{\mathbf{y}_i\}_{y=1}^{N}$) into K clusters: to this end, each observation is defined to belong to the cluster with the nearest mean, where each cluster mean (or center) serves as a prototype of the cluster.

Formally, given a set of observation $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N)$, where each observation is a d-dimensional real vector, K-means clustering aims at partitioning the N observations into K sets ($K \leq N$) $\{\mathcal{C}_1, \mathcal{C}_2, \ldots \mathcal{C}_K\}$, in order to minimize the following cost function, i.e. the within-cluster sum of squares (WCSS):

$$\text{cost}(\mathcal{C}_1, \ldots, \mathcal{C}_K; \mu_1, \ldots, \mu_2) = \sum_{j=1}^{K} \sum_{\mathbf{y} \in \mathcal{C}_j} \|\mathbf{y} - \mu_j\|^2 \;, \tag{5.11}$$

where μ_j is cluster center, i.e. the mean of the points contained in $\mathcal{C}_j$.

The problem of minimizing (5.11) is computationally difficult (NP-hard); however, there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum. Among these, the most common algorithm is the Lloyd's algorithm [80], which uses an iterative refinement technique. Given an initial set of K means $\mu_1^{(1)}, \ldots, \mu_K^{(1)}$, the algorithm proceeds by alternating between two steps:

1. *Assignment step*
   Assign each observation to the cluster whose mean yields the least WCSS. Since the sum of squares is the squared Euclidean distance, this is intuitively the "nearest" mean.
   $$\mathcal{C}_i^{(t)} = \left\{ \mathbf{y}_p : \|\mathbf{y}_p - \mu_i^{(t)}\|^2 \leq \|\mathbf{y}_p - \mu_j^{(t)}\|^2 \quad \forall j, 1 \leq j \leq K \right\} , \tag{5.12}$$
   where each $\mathbf{y}_p$ is assigned to exactly one cluster, even if it could be is assigned to two or more of them.
2. *Update step*
   Calculate the new means to be the *centroids* of the observations in the new clusters.
   $$\mu_i^{(t+1)} = \frac{1}{|\ \mathcal{C}_i^{(t)}\ |} \sum_{\mathbf{y}_j \in \mathcal{C}_i^{(t)}} \mathbf{y}_j \tag{5.13}$$
   Since the arithmetic mean is a least-squares estimator, this also minimizes the WCSS objective (5.11).

The algorithm converges when the assignments no longer change. Since both steps optimize the WCSS objective, and there only exists a finite number of such partitions, the algorithm must converge to a (local) optimum. There is no guarantee, though, that the global optimum is found using this algorithm.

In Section 2.3, we presented a new dictionary learning procedure for our NE-based SR problem, based on K-means clustering. With respect to K-means, however, we introduced two main changes:

- The clustering procedure is performed jointly on the LR and HR patches, to provide coherence between the two spaces, and
- The final dictionary is not composed by the cluster means, but an ad-hoc sampling of the clusters is done, by taking M elements from each partition.

5.3.2 K-SVD

K-SVD (from SVD - Singular value decomposition) is an algorithm for "designing overcomplete dictionaries for sparse representations" recently proposed by Aharon *et al.* [68].

More precisely, the task of K-SVD is to find the best dictionary of size K to represent the data samples $\{\mathbf{y}_i\}_{i=1}^N$ as sparse compositions, by solving

$$\min_{\mathbf{D},\mathbf{X}} \left\{ \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \right\} \quad \text{subject to} \quad \forall i,\ \|\mathbf{x}_i\|_0 \leq k\ . \tag{5.14}$$

The method proposed to solve (5.14) is an iterative method that alternates between sparse coding of the data samples based on the current dictionary, to update the matrix $\mathbf{X}$, and a process of updating the dictionary atoms to globally reduce the approximation error, that involves the computation of K SVD factorizations. The details procedure can be found in [68].

K-SVD can be seen a generalization of the K-means algorithm, where K-means is a special case of K-SVD with $k = 1$. In fact, as said in Section 5.3.1, the K-means algorithm can be viewed as a method to perform vector quantization (VQ). Given a set of input signals $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^N$, the clustering process partition the data into K clusters, each one identified by a mean. We can then see the set of the cluster means as a codebook of K codewords for VQ: each signal is represented by a single codeword according to a nearest neighbor assignment. The sparse representation problem addressed by K-SVD is then a generalization of the VQ objective, in which we allow each input signal to be represented by a linear combination of codewords, instead of a single one, which represent the dictionary atoms.

In the Section 5.3.5 we will present a novel algorithm in the spirit of K-SVD, i.e. where any input signal is sparsely represented by several elements of the dictionary. We add to the problem also a nonnegative constraint (i.e. the input data matrix as well as the dictionary matrix only contain nonnegative entries). We will, in fact, compare our algorithm to the nonnegative variant of K-SVD [89].

5.3.3 Nonnegative Matrix Factorization

Dictionary learning methods aim at finding a suitable representation of the data, namely a set of atoms that form a dictionary and possibly make particular structures present in the data explicit. When the learning process is performed by trying to adapt the dictionary to a set of signal examples $\mathbf{Y}$, dictionary learning can be seen mathematically as a matrix factorization problem, where the matrix $\mathbf{Y}$, containing column by column the input data vectors, is factorized into two other matrices such that $\mathbf{Y} \approx \mathbf{D}\mathbf{X}$: $\mathbf{D}$ is the learned dictionary; $\mathbf{X}$ is the representation matrix, in which each column represents the "projection" of the related input vectors w.r.t. the dictionary found.

Nonnegative Matrix Factorization (NMF) [74] refers to a few recent techniques designed to perform such factorization, with a nonnegative constraint on all the

factors. Therefore, NMF is in all respects a dictionary learning tool. Moreover, the nonnegative property makes it particularly attractive for image processing purposes, where, unless some transformations are applied on the image pixels, we have to deal with nonnegative values. The problem that NMF tries to solve is then the following one:

$$\mathbf{Y} \approx \mathbf{DX} \quad \text{s.t.} \quad \mathbf{D} \geq 0, \mathbf{X} \geq 0 \tag{5.15}$$

To solve (5.15), the same authors propose in [90] the following multiplicative rules to alternatively update $\mathbf{D}$ and $\mathbf{X}$

$$X_{ij} \leftarrow X_{ij} \frac{\left(\mathbf{D}^T\mathbf{Y}\right)_{ij}}{\left(\mathbf{D}^T\mathbf{DX}\right)_{ij}} , \tag{5.16}$$

$$D_{ij} \leftarrow D_{ij} \frac{\left(\mathbf{YX}^T\right)_{ij}}{\left(\mathbf{DXX}^T\right)_{ij}} , \tag{5.17}$$

where $(A)_{ij}$ indicates the element at the i-th row and the j-th column of the matrix $\mathbf{A}$. (5.17) and (5.16) are therefore element-wise multiplicative rules: each element of the matrices $\mathbf{D}$ and $\mathbf{X}$ is updated by multiplying itself by an appropriate factor. The above mentioned rules are shown in [90] to lead to a local minimum of the approximation error $\|\mathbf{Y} - \mathbf{DX}\|_F^2$.

NMF can be interpreted as a dictionary learning method. In fact, given a $d \times N$ data matrix $\mathbf{Y}$, we learn a dictionary matrix $\mathbf{D}$ of dimension $d \times K$, with $K \leq N$ usually, and a representation matrix $\mathbf{X}$. In other words, we can see each input data vector $\mathbf{y}_j$, a column of the matrix $\mathbf{Y}$, as a combination of the atoms of the dictionary, the columns of the matrix D, weighted by the coefficients of the respective column $\mathbf{x}_j$ in the matrix $\mathbf{X}$: $\mathbf{y}_j = \sum_{i=1}^{K} \mathbf{d}_i x_j(i)$.

5.3.4 Nonnegative matrix factorization with ℓ^0 sparsity constraints

In the previous section we presented NMF as a method to factor an input matrix into two matrices, with the only constraint that the two factors, as well as the input matrix, contain exclusively nonnegative entries. Now, we want to include in this formulation the concept of *sparse representation*, by imposing that each input vector $\mathbf{y}_j$ is represented by a combination of at most k atoms of the dictionary. This implies that a sparsity constraint is put on the representation matrix $\mathbf{X}$, according to the more natural measure of sparseness which is the ℓ^0 pseudo-norm: each column $\mathbf{x}_j$, which is the encoding of the related input vector $\mathbf{y}_j$, is constrained to have at most k nonzero elements.

Our NMF problem (5.15) then becomes the following sparse nonnegative matrix factorization problem:

$$\min_{\mathbf{D}\geq 0,\mathbf{X}\geq 0} \|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2 \quad \text{subject to} \quad \forall i \ \|\mathbf{x}_i\|_0 \leq k \ . \tag{5.18}$$

In the literature not many solutions have been presented to solve the problem (5.18), i.e. to find, for nonnegative input data, a nonnegative dictionary $\mathbf{D}$ and an exactly sparse encoding $\mathbf{X}$ w.r.t. the dictionary found. We take as references nonnegative K-SVD (NN K-SVD) [89], the nonnegative adaptation of K-SVD [68], the well-known method for dictionary learning, and the sparse NMF algorithm of Peharz *et al.* [92].

5.3.4.1 Nonnegative K-SVD

NN K-SVD, like K-SVD, consists of a 2-step procedure, which is summarized in Algorithm 5.3.1. Once the dictionary $\mathbf{D}$ is initialized (usually with random values or by taking K random vectors from the input data set), two steps are iterated until the desired number of iterations is reached: a *sparse nonnegative coding stage*, where the matrix $\mathbf{X}$ is computed, and a *dictionary update stage*, where the dictionary $\mathbf{D}$ is refined. As for the first step, a sparse coder referred to as *NNBP* is used. In NNBP, first the support of $\mathbf{X}$ is computed: the NMF formula for $\mathbf{X}$ (5.16) is iterated several times, to identify, for each input vector, the salient atoms that we will keep for the sparse representation. For a given column of $\mathbf{X}$, in fact, the k highest coefficients are considered to be referred to the important atoms; the remaining $(K-k)$, instead, are set to zero. The selected coefficients are then recomputed by solving a nonnegative least squares problem. The dictionary update is column-wise: for each dictionary atom $\mathbf{d}_j$, an error matrix is constructed, as the residual on $\mathbf{Y}$ when we exclude $\mathbf{d}_j$ from the dictionary $\mathbf{D}$. Then, the best rank-1 approximation for it is found through SVD. The factorization performed is used to update both $\mathbf{d}_j$ and the related row vector of $\mathbf{X}$ $\mathbf{x}_j^r$ (which reports the "contribution" of $\mathbf{d}_j$ for all the input vectors); therefore, in the second step of NN K-SVD $\mathbf{D}$ and $\mathbf{X}$ are contextually updated, as specified in Algorithm 5.3.1.

5.3.4.2 Sparse NMF

The sparse NMF algorithm of Peharz *et al.* [92] presents a "double-nested" scheme, as reported in Algorithm 5.3.2. After the dictionary initialization, the representation matrix $\mathbf{X}$ is computed with a sparse coder called by the authors *NMP* (Nonnegative Matching Pursuit), a nonnegative version of OMP [93], where, when choosing a new atom, we simply discard it if a negative projection results from that. Then, within an inner loop, $\mathbf{D}$ and $\mathbf{X}$ are iteratively refined by

Algorithm 5.3.1: NONNEGATIVE K-SVD [89](Y)

$\mathbf{D} \leftarrow$ initialization
repeat
  $\mathbf{X} \leftarrow$ sparsely code Y with D using NNBP with a ℓ^0 constraint
  $\mathbf{D}, \mathbf{X} \leftarrow$ column and row-wise updates with SVD
until $numIterations = M$

using the NMF multiplicative rules (5.17) and (5.16). Since the multiplications are element-wise, the sparsity of $\mathbf{X}$ is perfectly preserved (zeros remain). All this procedure is repeated several times: a new sparse encoding $\mathbf{X}$ is found using NMP, and the factorization is refined again in the NMF inner loop.

Algorithm 5.3.2: SPARSE NMF [92](Y)

$\mathbf{D} \leftarrow$ initialization
repeat
  $\mathbf{X} \leftarrow$ sparsely code Y with D using NMP
  repeat
    $\mathbf{D} \leftarrow$ update with (5.17)
    $\mathbf{X} \leftarrow$ update with (5.16)
  until $numIterations = M_2$
until $numIterations = M_1$

5.3.5 Proposed algorithm: K-WEB (K WEighted Barycenters)

In this section we present a new nonnegative dictionary learning method[1] to decompose an input data matrix of nonnegative signals into a dictionary of nonnegative atoms, and a representation matrix with a strict ℓ^0-sparsity constraint. We choose to pose a nonnegative constraint on the dictionary to learn, because, when dealing with pixel-based image patches or nonnegative features, it lets generate "meaningful" atoms. Moreover, non-negativity can sometimes be

1. The contribution related to the method described in this section appeared in [91].

a necessary condition (e.g. for the recovery of an originally nonnegative dictionary). The sparse constraint on the representation matrix, which makes each input vector expressible by a limited combination of atoms, instead, is according to the intuition that images and image patches have been shown to follow sparse models in opportune domains.

The method has the same goal as the algorithms described in Section 5.3.4. As in NN K-SVD and sparse NMF, we want to solve the problem (5.18), i.e. learn a nonnegative dictionary suitable for exactly sparse representations, by finding a factorization of a signal example matrix $\mathbf{Y}$. For this sake, we want to design two methods to update, possibly separately, the two factors:

- a *nonnegative sparse coding stage* to (re-)compute $\mathbf{X}$;
- a *dictionary update stage* to update $\mathbf{D}$.

Nonnegative sparse coding In the nonnegative sparse coding stage, we aim at finding, for each input vector $\mathbf{y}_j$, the best nonnegative ℓ^0 - sparse coding $\mathbf{x}_j$ w.r.t to a given dictionary:

$$\mathbf{x}_j = \arg\min_{\mathbf{x}\geq 0} \|\mathbf{y}_j - \mathbf{D}\mathbf{x}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{x}\|_0 \leq k \; . \tag{5.19}$$

Each vector found replace the related column in X.

To solve (5.19), we decide to use the NMP algorithm described in [92]. We believe, indeed, that a greedy "OMP-like" choice of the atoms leads to a better sparse approximation than the NNBP method used in NN K-SVD. In the latter, the atoms, that have the highest coefficients according to a first NMF approximation of $\mathbf{X}$, are chosen, but these atoms are not necessarily those ones which minimize the approximation error. We also believe that the sparse coding stage, and so the computation of the support of $\mathbf{X}$, should be performed as much frequently as possible. Therefore, we choose to adopt the simple 2-step scheme of NN K-SVD (Algorithm 5.3.1), where the identification of the support of $\mathbf{X}$ is done at each step of the unique loop, rather than the double-nested one of [92].

Dictionary update with *K-WEB* As for the second step of our algorithm (update of the dictionary), we propose an original way to update $\mathbf{D}$ column by column, by considering $\mathbf{X}$ fixed and not updating it as well, as done in K-SVD. $\mathbf{X}$ and $\mathbf{D}$ are therefore updated separately in the two steps of the algorithm.

By considering the error matrix $\mathbf{E} = \mathbf{Y} - \mathbf{D}\mathbf{X}$, we can separate the contribution to the matrix approximation due to a particular dictionary atom $\mathbf{d}_j$ in this way:

$$\mathbf{E} = \left(\mathbf{Y} - \sum_{i \neq j} \mathbf{d}_i \mathbf{x}_i^r \right) - \mathbf{d}_j \mathbf{x}_j^r = E_{\cdot j} - \mathbf{d}_j \mathbf{x}_j^r \; , \tag{5.20}$$

where $E_{\cdot j}$ is the error matrix without considering the contribution of the atom $\mathbf{d}_j$.

As our goal is to minimize the norm of $\mathbf{E}$, we would like $E_{\cdot j} \approx \mathbf{d}_j \mathbf{x}_j^r$. In NN K-SVD the matrix $E_{\cdot j}$, once pruned from those columns for which the related coefficients in $\mathbf{x}_j^r$ are zero, is factorized according to SVD. However, SVD, as it performs a full factorization, forces us to update both $\mathbf{d}_j$ and $\mathbf{x}_j^r$, which are placed respectively in $\mathbf{D}$ and $\mathbf{X}$. We want, instead, to find $\mathbf{d}_j$, by considering $\mathbf{x}_j^r$ fixed. The problem to solve is the following:

$$\mathbf{d}_j^\star = \arg\min_{\mathbf{d}_j \geq 0} \|E_{\cdot j} - \mathbf{d}_j \mathbf{x}_j^r\|_2^2 \,. \tag{5.21}$$

The problem (5.21) is in a quadratic form in any of the coordinates of $\mathbf{d}_j$, and has the following closed-form solution:

$$\mathbf{d}_j^\star = \max\left(\frac{\sum_{i=1}^N E_{\cdot j,i} x_j^r(i)}{\mathbf{x}_j^r \mathbf{x}_j^{rT}}, \mathbf{0} \right) , \tag{5.22}$$

where $E_{\cdot j,i}$ is the i-th column of the matrix $E_{\cdot j}$, and the max operator is intended element-wise.

The minimization problem described in (5.21) has a geometric interpretation (see Figure 5.1). Let us first consider that $\mathbf{x}_j^r$ is a vector of all ones. $\mathbf{d}_j^\star$ in (5.21) is then the vector that minimizes the sum of all the Euclidean distances, between itself and the column vectors of $E_{\cdot j}$, i.e. the barycenter of the related set of points. With a given vector of coefficients $\mathbf{x}_j^r$, the solution is the *weighted barycenter* expressed by (5.22). Therefore, we call this new method for updating the dictionary, which requires the computation of K WEighted Barycenters, *K-WEB*.

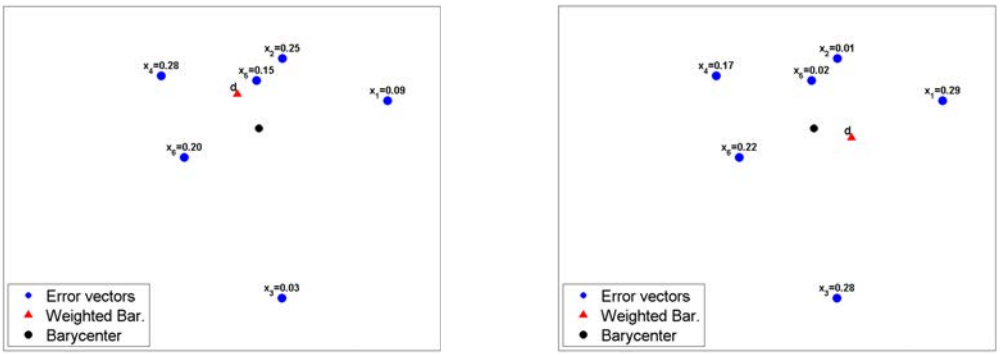


Figure 5.1: Geometrical interpretation of the computation of the weighted barycenter - While the real barycenter stays the same, the weighted barycenter moves according to the different values of the data points.

The procedure we design is finally summarized in Algorithm 5.3.3. It is a 2-step procedure: NMP is used as a sparse coder to compute $\mathbf{X}$; the new K-WEB method is subsequently used to update the dictionary matrix $\mathbf{D}$.

Algorithm 5.3.3: K-WEB ALGORITHM(Y)

$\mathbf{D} \leftarrow$ initialization
repeat
  $\mathbf{X} \leftarrow$ sparsely code Y with D using NMP
  $\mathbf{D} \leftarrow$ column-wise update with K-WEB
until $numIterations = M$

5.3.6 Comparison between nonnegative dictionary learning methods for sparse representations

To validate our new nonnegative dictionary learning method with ℓ^0 constraints, presented in the previous section, we apply it to two different scenarios in image processing, and compare with the other similar methods described in Section 5.3.4.

5.3.6.1 Image patch approximation

The first application is an image patch approximation problem: we want to train a dictionary suitable for sparsely approximating natural image patches. To do that we take $N = 50000$ 8×8 patches (therefore the dimension of the input data vectors $d = 64$), randomly taken from natural images. The so formed matrix $\mathbf{Y}$ is the input of the dictionary learning algorithm, which factorizes it into a dictionary $\mathbf{D}$ and a representation matrix $\mathbf{X}$. The number of atoms chosen is $K = 250$; the target sparsity level is $k = 15$, i.e. each column of $\mathbf{X}$ contains at most 15 nonzero entries. After the dictionary learning procedure, $\mathbf{D}$ is used with other test images: the appropriate sparse coder (NNBP for NN K-SVD, NMP for our proposed algorithm and the sparse NMF method of Peharz *et al.*) is used to find a sparse approximation of each input patch of the test image, by taking the atoms of $\mathbf{D}$ as bases. Table 5.1 reports the performance (*p-index*) of the three methods with a set of test images different from the training set, in terms of average MSE (*avgMSE*) of the patch reconstructions converted in a PSNR-like

logarithmic scale:

$$\begin{aligned} avgMSE &= \frac{1}{N_t \cdot d} \sum_{i=1}^{N_t} \|\mathbf{y_i} - D\mathbf{x_i}\|_2^2 \quad , \\ p\text{-}index &= 10 \log_{10} \left(avgMSE^{-1} \right) \end{aligned} \tag{5.23}$$

where N_t is the total number of test patches, $\mathbf{y}_i$ is a particular patch, and $\mathbf{x}_i$ is its sparse representation w.r.t. $\mathbf{D}$.

	Images			
Method	**Bird**	**Butterfly**	**Eyetest**	**Head**
NN K-SVD	25.54	24.69	21.18	27.10
Sparse NMF	40.83	32.13	23.32	37.54
K-WEB	**44.25**	**36.44**	**26.20**	**40.99**

Table 5.1: Results of *K-WEB* and other methods on the patch approximation problem - The results are for different test images.

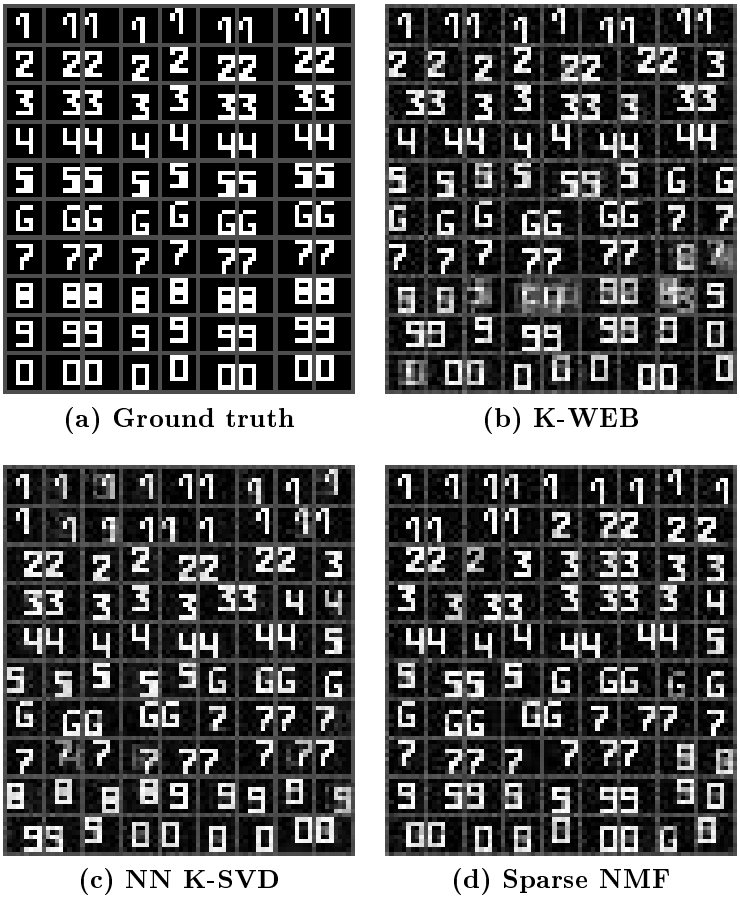
As we can seen from the table, our method sensitively outperforms the two methods in the literature.

5.3.6.2 Dictionary recovery

The second application is a dictionary recovery problem: we have an original dictionary $\mathbf{D}$, with which we generate a data matrix $\mathbf{Y}$, by sparsely combining atoms of it randomly taken (a random sparse matrix $\mathbf{X}$ is generated); some Gaussian noise is also added. Thus, $\mathbf{Y} \approx \mathbf{DX}$. $\mathbf{Y}$, used as an input of the dictionary learning process, is factorized in a dictionary $\hat{\mathbf{D}}$ and a coding matrix $\hat{\mathbf{X}}$, which we compare with the original ones. The goal of this test is to see if the method is suitable for recognizing truly sparse data and possibly retrieve the original dictionary that was actually used to generate them.

The original dictionary is depicted in the first subplot of Fig. 5.2, and consists of $K = 90$ 8×8 patches, then vectored, representing the 10 digits and shifted versions of them. The number of patches generated for Y is $N = 2000$; $k = 5$.

Table 5.2 reports the results of the recovered dictionaries for the tree methods analyzed: NN-KSVD, the sparse NMF method of Peharz *et al.*, and our proposed algorithm. The first parameter presented is the training error $t\text{-}err = \frac{1}{N \cdot d} \|\mathbf{Y} - \hat{\mathbf{D}}\hat{\mathbf{X}}\|_2^2$, which measure the goodness of the training process itself. The other two parameters better indicate the performance of the test made. The ground truth (GT) error measure the distance between $\hat{\mathbf{D}}$ and $\mathbf{D}$: for each atom of the original dictionary $\mathbf{d}_j$, we find the closest atom in $\hat{\mathbf{d}}_j$ and measure the reciprocal distance ($\text{dist}_j = 1 - |\mathbf{d}_j \cdot \hat{\mathbf{d}}_j|$); the *GT error* is the sum of all these



(a) Ground truth **(b) K-WEB**

(c) NN K-SVD **(d) Sparse NMF**

Figure 5.2: Results of *K-WEB* and other methods on the dictionary recovery problem - The original dictionary consisting of 90 figures and the dictionaries recovered are shown.

minimum distances. The percentage of well recovered atoms in another indicator for the success of the test: an atom of $\mathbf{D}$, $\mathbf{d}_j$, is considered well recovered if we find in $\hat{\mathbf{D}}$ a vector distant up to a certain threshold ($\text{dist}_j < 0.01$).

Method	Train. error	GT error	% recovered
NN K-SVD	0.0139	1.557	61.11 %
Sparse NMF	0.0129	1.391	63.33 %
K-WEB	**0.0135**	**1.407**	**71.11** %

Table 5.2: Results of *K-WEB* and other methods on the dictionary recovery problem - The training error and other two metrics on the recovery fidelity are taken into account.

Table 5.2 shows that the method of Peharz *et al.* gives a slightly better GT er-

ror, but our proposed method is able to better recover a higher number of original digits. Namely, 64 digits out of 90 ($71.\bar{1}\%$) in the case of our algorithm, whereas the sparse NMF method recovers 57 digits out of 90 ($63.\bar{3}\%$). This is confirmed in Fig. 5.2, that shows as images also the recovered dictionaries. With respect to the other methods, our K-WEB algorithm, despite having some problems with certain digits, leads to more accurate recovered patches; the distribution of the digits is also better "balanced".

Chapter 6

Conclusions and perspectives

The work of this thesis focused on the study of super-resolution (SR) as a technique to augment the spatial resolution, i.e. the total number of independent pixels, of images and videos, to a greater extent than traditional interpolation methods. In particular, we adopted the example-based single-image SR approach, whose aim is to create a high-resolution (HR) output image from as little as a single low-resolution (LR) input image, by means of a dictionary of correspondences of LR and HR patches (i.e. the "examples"). The dictionary is meant to bring extra information to enrich the original LR image with high-frequency content when upscaling it. The SR procedure follows the machine learning paradigm, where the HR output image is predicted/estimated patch by patch: for each LR input patch we compute a model on the basis of "local examples" (the closest LR patches in the dictionary and the corresponding HR patches), and we use this model to predict the related HR output patch.

The procedure described is quite general and leaves room for different kinds of investigations. Among the aspects that can be analyzed we have in fact:

- How to represent the patches (i.e. which features to use), in order to make the learning process the most effective possible,
- The learning method used for the SR single reconstructions, i.e. how each HR output patch is predicted, given a LR input patches and a set of training examples (pairs of LR-HR patches), and
- The method to construct the dictionary.

In the study conducted, we investigated all these aspects, by having as objective either the quality of the finally super-resolved images or a trade-off between the latter and the computational complexity of the algorithm. This led us to the formulation of the different algorithms for upscaling single images that we presented in Chapter 2 and Chapter 3. In Chapter 4, instead, we dealt with the problem of upscaling a whole video sequence, by equally providing new algorithms and analyses. Finally, in Chapter 5, we proposed a classification of the

SR techniques considered (patch reconstruction and dictionary learning methods) within the framework of classical learning methods.

In Section 6.1, we detail and comment, chapter by chapter, all the contributions presented. Subsequently, in Section 6.2, we discuss some possible future directions of research related to the work done.

6.1 Summary of contributions

In Chapter 1 we formally defined the SR problem and classified the many different methods according to their characteristics and the approaches adopted. In particular, for example-based single-image SR we distinguished methods employing an external dictionary and methods that make use of an internal dictionary.

In Chapter 2, then, we presented our work on external-dictionary single-image SR, by using neighbor embedding (NE) as patch reconstruction method. Starting from the original NE-based SR algorithm of Chang *et al.* [49], we introduced several modifications that led to different algorithms showing progressive improvements. The algorithm of Chang *et al.* is based on a weight computation method derived from the Locally Linear Embedding (LLE) method [63, 70], where the weights of a patch reconstruction sum up to one, and uses gradient-based features to represent the LR patches. By performing an analysis of the LLE weights with different feature configurations and by varying the number of neighbors K (see Section 2.2.1), we observed that simple mean-removed luminance values (centered features) were the best performing features, except for a drop in the performance occurring for a value of K equal to the dimension of the LR vectors. We explained this with the fact that the weights, computed to approximate each LR input patch with a supplementary equality constraint, were "too fitted" on the LR data. We then proposed two main changes to the original algorithm:

- To use a non-negativity inequality constraint to "relax" the weight computation,
- To use centered features, instead of gradient-based values, to represent both the LR and HR patches.

The algorithm resulting from these changes is a nonnegative NE-based SR algorithm that we called "NoNNE" and presented in Section 2.2. The NoNNE algorithm showed good *PSNR* performance, strictly increasing with the number of neighbors: compared to other state-of-the-art methods, it presented better quality results than other single-pass algorithms using an external dictionary, whereas the comparison was slightly unfavorable in the case of the well-known multi-pass algorithm of Glasner *et al.* [58] employing an internal dictionary constructed by means of a pyramid of "self-produced". As for the computational complexity, the NoNNE algorithm had clearly the best performance (i.e. lower running times),

thanks to the employment of the "low-cost" centered features. The low-complexity feature, combined with the good quality of the super-resolved results, made this algorithm particularly attractive for real-time tasks. As a matter of fact, the business unit of our industrial partner, Alcatel-Lucent, decided to implement the NoNNE algorithm to possibly include it in some of their applications.

The rest of Chapter 2 aimed at filling the performance gap between algorithms using external dictionaries and those ones, like the pyramid-based algorithm of Glasner *et al.*, employing internal dictionaries. A problem observed for the former is that the performance of the external dictionaries are often image-dependent. A fixed dictionary may not be suitable for any input image; in order to make it work fairly well with a variety of inputs, we have to include a large amount of different-content image patches, thus ending up with big unsustainable dictionary sizes. To deal with this problem, we then proposed in Section 2.3 a new procedure to construct an external dictionary, that takes as input a big "all-inclusive" dictionary and subsequently extracts from it relevant pairs of patches. The procedure consists of two original steps:

- A joint k-means cluster (JKC) stage, that jointly clusters the sets of LR and HR patches by following a neighborhood preservation criterion, and at the same time already discards some "bad" pairs of patches, and
- A sampling stage, to intelligently select the most salient patches.

The tests made, by using the NoNNE algorithm as a base, proved the goodness of the proposed procedure: the newly built dictionary, while presenting a reduction in size of a factor of 5, gave even better performance than the big dictionary and performed generally well with different input images.

A third contribution presented in Chapter 3 concerned the idea of "enhanced interpolation", that uses the iterative back-projection (IBP) algorithm to enhance the results of an interpolation process. The new enhanced interpolation procedure can be used wherever in the algorithm an interpolation step is requested, i.e. at the beginning of the algorithm, to obtain a better guess of the HR output image, and in the training phase. We then proposed and evaluated three different training schemes to generate the dictionary of patches. The winning one, involving, indeed, enhanced interpolation and the computation of high-frequency-residual images (the JKC-based procedure is also performed to further improve the dictionary), laid the basis for a novel NE-based SR algorithm that we called "NEEB" (Neighbor Embedding using Enhanced Bicubic interpolation). The NEEB algorithm further improved the results of our previous algorithms, by showing in this case comparable performance w.r.t. the pyramid-based algorithm of Glasner *et al.* [58] and thus finally bridging the gap between external and internal-dictionary methods.

Chapter 3 dealt instead with our contribution as for example-based SR using

an internal dictionary. Our goal in this case was to improve the method of Glasner *et al.*. In [58], NE is employed to combine the self-examples selected throughout the pyramid. We believed instead that, given a LR input patch, since the neighbors that can be found in the pyramid are more relevant to it than any patches we can ever find in an external dictionary, they are particularly suitable to learn a mapping function to directly map it to the corresponding HR patch. The use of direct mapping (DM) instead of NE and a new original scheme based on a double pyramid, instead of the single pyramid of Glasner *et al.* led us to the definition of a new algorithm, whose results turned out to be better than any other state-of-the-art methods considered.

In Chapter 4 we then considered the video case, i.e. the upscaling of a whole video sequence, and provided a systematic analysis of two scenarios: a *Scenario A*, where LR frames are interlaced with periodic HR "key frames", and a *Scenario B*, where the video sequence is composed only of LR frames. Our contributions to this investigation mainly included:

- The introduction of a new weight computation method, still based on a least squares approximation problem with a non-negativity constraint, but also containing additional terms to enforce temporal consistency between frames,
- The comparison of different procedures adapting the single-image SR methods presented in the previous chapters, and
- An analysis in the coding context, where an ad-hoc SR approach to video coding is compared with direct encoding of the original video sequence.

Interesting remarks have been made in particular in the case of Scenario A, where an effective video SR procedure, using an internal dictionary built from the key frames, has been designed. In this case, too, the SR approach to video coding showed promising results, especially for low coding rates: as a matter of fact, the SR algorithm, applied to highly compressed video frames firstly decoded, is able to smooth out some of the compression artifacts, thus producing globally more pleasant images.

Chapter 5, finally, presented more theoretical topics, by taking a look at some techniques related to the machine learning area and placing them within the framework of the SR methods previously debated. Within the family of unsupervised learning techniques, we also proposed a novel algorithm, that we called *K-WEB* (the acronym stands for K-WEighted Barycenters), to perform the learning of a dictionary of non-negative atoms suitable for sparse representations. When comparing the proposed algorithm to other similar methods present in the literature, we observed better results in two different problems considered.

6.2 Comments and perspectives

In this thesis we examined the problem of single-image SR upscaling thoroughly, by presenting different algorithms and procedures belonging to the family of example-based SR. Specifically, we designed three single-image SR algorithm which we are quite proud of, each one presenting its own pros:

- The NoNNE algorithm, that represents a good trade-off between quality and low-complexity,
- The NEEB algorithm (that also makes use of the JKC-based dictionary construction procedure), which is one the most performing methods among those ones using an external dictionary and has comparable performance than other multi-pass algorithms based on internal dictionaries, and
- The double-pyramid algorithm, for internal dictionary, whose target is to maximize the quality of the super-resolved images.

While the multi-frame "branch" was particularly prosperous in the previous decade, in the recent years, many algorithms have been developed for the single-image SR problem, including a large variety of procedures that already show very promising results. We then believe that the attention to this topic should be mostly directed from the conception of new procedures to other important aspects, like the theoretical validation and their practical implementation. As for the former, SR procedures, especially example-based methods, are often the result of an heuristic approach (consider, for example, the hypothesis of manifold similarity of the LLE-based algorithm, not fully supported by the evidence). We then think that some extra studies can be conducted to completely unveil the connection between example-based SR and machine learning, as we partially did in Chapter 5, and thus provide a better theoretical framework to the latter. Another theoretical aspect to clarify is the use of the image generation model, which considers a blurring stage (where the blur operator is often considered to be Gaussian) and a down-sample operation. In the majority of the algorithm in the literature, the model is assumed to be known; an analysis on how to estimate the parameters of the model directly from a generic LR image taken as input would be therefore useful. As for the practical implementation of single-SR methods, we believe that there is a clear room for single-image SR algorithms to be optimized (e.g. the example-based SR procedure consists of single patch reconstructions that could be carried out via parallelized tasks, thus achieving a considerable speed-up), and finally employed as fast upscaling methods in the same way as analytic interpolation methods.

While the multiple-input single-output (MISO) and single-input single-output (SISO) SR problems have been largely discussed in the literature, giving rise to, respectively, multi-frame and single-image SR methods, the multi-input multi-

output (MIMO) or video-to-video SR problem is not totally explored and new procedures can be certainly conceived. In Chapter 4 we dealt with the MIMO problem, by basically adopting a single-image strategy. This approach proved to be worthwhile in the case of Scenario A, where we could build an effective dictionary from the key frames: also by performing an analysis in the video coding context, SR has been proved, even surprisingly, to be a possibly useful tool for compression in cooperation with the extremely efficient HEVC codec (further studies in this sense are welcome). Scenario B, however, highlighted the necessity to design new procedures to enhance the video upscaling, by considering several frames at time in a multi-frame fashion (this could also optimize the time-complexity of the algorithm.

Finally, we think that there are still a few applications in which SR could be used. An example is given by the plenoptic images that can be taken by some recently commercialized fight-field camera, which consist in a series of small images referring to different viewpoints. A drawback of such systems is in fact the low resolution that the final images have.

Appendix A

Proof of the SUM1-LS solution

In this appendix we provide a proof for the solution to the SUM1-LS problem expressed in (2.4), appearing in the weight computation step of the LLE algorithm [70] and adopted also in the original neighbor embedding SR algorithm of Chang *et al.* [50]. The problem aims at finding the best linear combination of neighbors that approximates a given input data point, with the condition that the weights sum up to one. In the SR problem, we speak about patches (our data points): the weights computed represent in this case a linear combination of LR patches in a dictionary, which are selected in order to approximate a given LR patch of the LR input image.

For each LR input patch $\mathbf{x}_i^l$ we then define a reconstruction error as (A.1a), and find the weights by minimizing it, subject to the above-mentioned sum-to-one condition, i.e. $\sum_j w_{ij} = 1$.

The reconstruction error for the data point $\mathbf{x}_i^l$ is so defined:

$$\varepsilon_i = \|\mathbf{x}_i^l - \sum_{\mathbf{y}_j^l \in \mathcal{N}_i} w_{ij}\mathbf{y}_j^l\|^2 \tag{A.1a}$$

$$= \|\mathbf{x}_i^l \sum_{\mathbf{y}_j^l \in \mathcal{N}_i} w_{ij} - \sum_{\mathbf{y}_j^l \in \mathcal{N}_i} w_{ij}\mathbf{y}_j^l\|^2 \tag{A.1b}$$

$$= \| \sum_{\mathbf{y}_j^l \in \mathcal{N}_i} w_{ij}(\mathbf{x}_i^l - \mathbf{y}_j^l)\|^2 \tag{A.1c}$$

$$= \|(\mathbf{x}_i^l \mathbf{1}^{\mathrm{T}} - \mathbf{Y}_i^l)\mathbf{w}_i\|^2 \tag{A.1d}$$

$$= \mathbf{w}_i{}^{\mathrm{T}}(\mathbf{x}_i^l \mathbf{1}^{\mathrm{T}} - \mathbf{Y}_i^l)^{\mathrm{T}}(\mathbf{x}_i^l \mathbf{1}^{\mathrm{T}} - \mathbf{Y}_i^l)\mathbf{w}_i \tag{A.1e}$$

$$= \mathbf{w}_i{}^{\mathrm{T}}\mathbf{G}_i\mathbf{w}_i \tag{A.1f}$$

where

- $\{\mathbf{y}^l_{j_1}, \ldots, \mathbf{y}^l_{j_K}\} = \mathcal{N}_i$
- $\mathbf{Y}^l_i = \left[\ \mathbf{y}^l_{j_1} \ \middle|\ \ldots \ \middle|\ \mathbf{y}^l_{j_K}\ \right]$: $D \times K$ matrix (all neighbors of $\mathbf{x}^l_i$ as columns)
- $\mathbf{w}_i = \begin{bmatrix} w_{ij_1} \\ \vdots \\ w_{ij_K} \end{bmatrix}$: $K \times 1$ vector of the weights
- $\mathbf{x}^l_i \mathbf{1}^{\mathrm{T}} = \left[\ \mathbf{x}^l_i \ \middle|\ \ldots \ \middle|\ \mathbf{x}^l_i\ \right]$: $D \times K$ matrix
- $(\mathbf{x}^l_i \mathbf{1}^{\mathrm{T}} - \mathbf{Y}^l_i) = \left[\ \mathbf{x}^l_i - \mathbf{y}^l_{j_1} \ \middle|\ \ldots \ \middle|\ \mathbf{x}^l_i - \mathbf{y}^l_{j_K}\ \right]$: $D \times K$ matrix
- $\mathbf{G}_i := (\mathbf{x}^l_i \mathbf{1}^{\mathrm{T}} - \mathbf{Y}^l_i)^{\mathrm{T}} (\mathbf{x}^l_i \mathbf{1}^{\mathrm{T}} - \mathbf{Y}^l_i)$: $K \times K$ matrix.

Minimizing (A.1f) leads us to solve a *Constrained Least Squares* problem, subject to the constraint $\mathbf{1}^{\mathrm{T}} \mathbf{w}_i = 1$. The method of Lagrange multipliers can be used to find a solution to that.

The Lagrangian functional is expressed as follows:

$$J_i = {\mathbf{w}_i}^{\mathrm{T}} \mathbf{G}_i \mathbf{w}_i + \lambda(\mathbf{1}^{\mathrm{T}} \mathbf{w}_i - 1) \ . \tag{A.2}$$

We then compute the gradient of J_i with respect to the weight vector and the Lagrange multiplier:

$$\frac{\partial J_i}{\partial \mathbf{w}_i} = 2\mathbf{G}_i \mathbf{w}_i + \mathbf{1}\lambda = 0 \tag{A.3}$$

$$\frac{\partial J_i}{\partial \lambda} = \mathbf{1}^{\mathrm{T}} \mathbf{w}_i - 1 = 0 \tag{A.4}$$

$$\text{(A.3)} \quad \longrightarrow \quad \mathbf{w}_i = -\frac{1}{2} {\mathbf{G}_i}^{-1} \mathbf{1}\lambda$$

$$\text{(A.4)} \quad \longrightarrow \quad \mathbf{1}^{\mathrm{T}} \left(-\frac{1}{2} {\mathbf{G}_i}^{-1} \mathbf{1}\lambda \right) = 1 \quad \Longrightarrow \quad \lambda = \frac{-2}{\mathbf{1}^{\mathrm{T}} {\mathbf{G}_i}^{-1} \mathbf{1}}$$

$$\Longrightarrow \quad \mathbf{w}_i = \frac{\mathbf{G}_i^{-1} \mathbf{1}}{\mathbf{1}^{\mathrm{T}} \mathbf{G}_i^{-1} \mathbf{1}} \ . \tag{A.5}$$

The solution, as written in (A.5), requires an explicit inversion of the matrix $\mathbf{G}_i$. If this matrix is singular or nearly singular (as arises, for example, when the chosen number of neighbors is larger than the input dimension), it can be conditioned by adding a small multiple of the identity matrix, e.g.

$$\mathbf{G}_i = \mathbf{G}_i + \mathbf{I}_K \times \Delta \times \mathit{trace}(\mathbf{G}_i) \times K \ , \qquad \text{with } \Delta \ll \mathit{trace}(\mathbf{G}_i) \ .$$

Résumé en français

Introduction

Les techniques de traitement du signal pour augmenter la qualité visuelle d'images et de vidéos sont aujourd'hui particulièrement intéressantes. Une première raison de cette affirmation est due au progrès technologique qui a élevé le niveau et les attentes des utilisateurs en bénéficient qui bénéficiant de contenus multimédias. En fait, la dernière décennie, en fait, a témoigné d'une révolution dans la technologie d'affichage en grand format pour l'utilisateur final : les marchés de consommation sont actuellement inondés de télévisions et autres systèmes d'affichage, qui présentent des images de très haute qualité à des résolutions spatiales et temporelles très élevées.

Toutefois, malgré l'intérêt croissant à leur égard, les contenus de haute qualité ne sont pas toujours disponibles pour être affichés. Les sources de vidéos sont malheureusement souvent d'une qualité inférieure à celle souhaitée, en raison de plusieurs causes possibles : du sous-échantillonnage spatial ou temporel qui peut être nécessaire, d'une dégradation à cause du bruit, d'une haute compression, etc. En outre, les nouvelles sources de contenus vidéo, comme Internet ou les appareils mobiles, ont généralement une moins bonne qualité d'image que la diffusion conventionnelle d'émissions télévisées.

En dehors de l'expérience utilisateur, les raisons de la nécessité d'augmenter la résolution d'une vidéo ou d'une image peuvent également être imposées par le contexte particulier d'application considéré. En effet, de nombreuses applications, par exemple la surveillance vidéo et la télédétection, nécessitent l'affichage d'images à haute résolution, éventuellement pour des tâches spécifiques telles que la reconnaissance d'objets ou des opérations de zoom-in.

La Super-résolution (SR) s'adresse spécifiquement aux problèmes mentionnés ci-dessus, puisqu'elle se réfère à une famille de procédures qui visent à augmenter la résolution, et donc la qualité, d'images données, dans une plus grande mesure que les algorithmes de traitement d'image classiques. Différemment des méthodes traditionnelles (l'interpolation, l'aiguisage d'images, etc.) , l'objectif de la SR est plus ambitieux : le méthodes de SR, en fait, visent à estimer des détails

en haute résolution (HR) qui ne sont pas présents dans l'image originale, en ajoutant de nouvelles plausibles hautes fréquences. Pour poursuivre cet objectif, deux approches principales ont été étudiées dans la littérature dans ces dernières années : la SR "multi-frame" et la SR "single-image".

Les méthodes multi-frame comptent sur la présence de multiples images liées à la même scène : ces multiples images sont fusionnées ensemble de façon appropriée pour former une seule image de sortie en HR. Les méthodes single-image représentent en quelque sorte un défi encore plus grand, car il s'agit de créer de nouvelles informations en haute fréquence à partir d'une seule image d'entrée. Parmi les méthodes de SR single-image, une catégorie importante est représentée par des algorithmes qui utilisent des techniques d'apprentissage automatique. L'apprentissage automatique peut être généralement considéré comme cette branche de l'intelligence artificielle qui concerne la construction et l'étude d'algorithmes capables "d'apprendre à partir des données". Ses origines remontent à il y a plusieurs décennies, mais c'est seulement dans les années quatre-vingt-dix qu'il est devenu particulièrement populaire. Depuis lors, beaucoup d'algorithmes ont été développés pour résoudre différents problèmes dans une variété de domaines scientifiques.

Intéressés par l'approche de la SR pour augmenter la résolution d'une image, et intrigués par l'efficacité des techniques d'apprentissage automatique, nous avons pendant ce doctorat surtout étudié le problème de la SR et l'application de méthodes d'apprentissage automatique à cette fin. En particulier, nous avons adopté une approche single-image "basée exemples", où une seule image basse résolution (BR) est agrandie au moyen d'un dictionnaire d'exemples. Ce dictionnaire d'exemples, qui dans ce cas consiste en des correspondances de "patchs" d'images BR et HR, vu dans le cadre de l'apprentissage automatique, représente les données desquelles nous voulons "apprendre".

Le reste de ce manuscrit est structuré comme suit. Nous commençons avec le Chapitre 1 où nous donnons un aperçu général de la SR, en classant les méthodes multi-frame et les méthodes single-image, et discutons de la pertinence de ces travaux. Les chapitres 2 et 3 présentent nos contributions au sujet de la SR single-image, décrivant de nouveaux algorithmes utilisant, respectivement, des dictionnaires externes et internes. En particulier, les méthodes basées sur un dictionnaire externe, présentées dans le Chapitre 2 sont le résultat de plusieurs éléments qui ont amené à la formulation de trois nouveaux algorithmes, décrits dans des publications distinctes. L'extension de la SR au cas de la vidéo est présentée dans le Chapitre 4, où nous considérons deux différents scénarios (selon la disponibilité d'images HR périodiques).

Différentes procédures de SR (les méthodes single-image décrites dans les chapitres précédents, adaptés au cas de la vidéo, ainsi que de nouvelles procédures) sont comparées, et une analyse dans le contexte de la compression vidéo

est également effectuée. Dans le Chapitre 5, nous proposons ensuite un recueil de certains techniques d'apprentissage automatique que nous avons étudiées durant ce doctorat, en mettant un accent particulier sur leur application dans les méthodes de SR considérées. Enfin, nous terminons cette thèse en résumant nos accomplissements, nous tirons des conclusions et nous discutons de possibles orientations futures.

Chapitre 1 : Super-résolution, définition du problème et revue de l'état de l'art

Dans le Chapitre 1, nous présentons les principes fondamentaux de la super-résolution, en fournissant une définition générale et en la classifiant.

Selon le schéma taxonomique de la Figure A.1, les algorithmes de SR peuvent être classés en fonction du nombre d'images d'entrée et d'images de sortie participant au processus.

- Algorithmes SISO (single-input single-output), quand une seule image HR est produite à partir d'une seule image BR,
- Algorithmes MISO (multiple-input single-output), quand nous nous occupons de l'intégration de plusieurs images BR pour estimer une unique image HR, et
- Algorithmes MIMO (multiple-input multiple-output), également connus comme SR vidéo-à-vidéo, quand nous essayons de sur-échantillonner une séquence vidéo entière.

Une autre classification peut être réalisée selon les méthodologies réellement utilisées. Dans ce cas-là, on peut distinguer deux grandes catégories de méthodes : les méthodes multi-frame et les méthodes single-image. Pour ces deux familles d'algorithmes, les principes et les principales méthodes de l'état de l'art sont décrits. Parmi les méthodes de SR single-image, une famille particulière de méthodes qui prend le nom de "SR basée exemples", et sur laquelle la plupart du travail de cette thèse se concentre, est ensuite analysée.

La SR basée exemples vise à agrandir une image d'entrée BR au moyen d'un dictionnaire d'exemples qui exprime localement la relation entre une image BR et son homologue HR. Les exemples utilisés sont généralement sous la forme de "patchs", c'est à dire des blocs carrés de pixels (par exemple 3×3 ou 5×5). Le dictionnaire est donc une collection de blocs qui, deux par deux, forment des paires. Une paire se compose précisément d'un bloc BR et de sa version HR, qui contient des détails de haute fréquence. Les algorithmes de SR basés exemples se composent de deux phases :

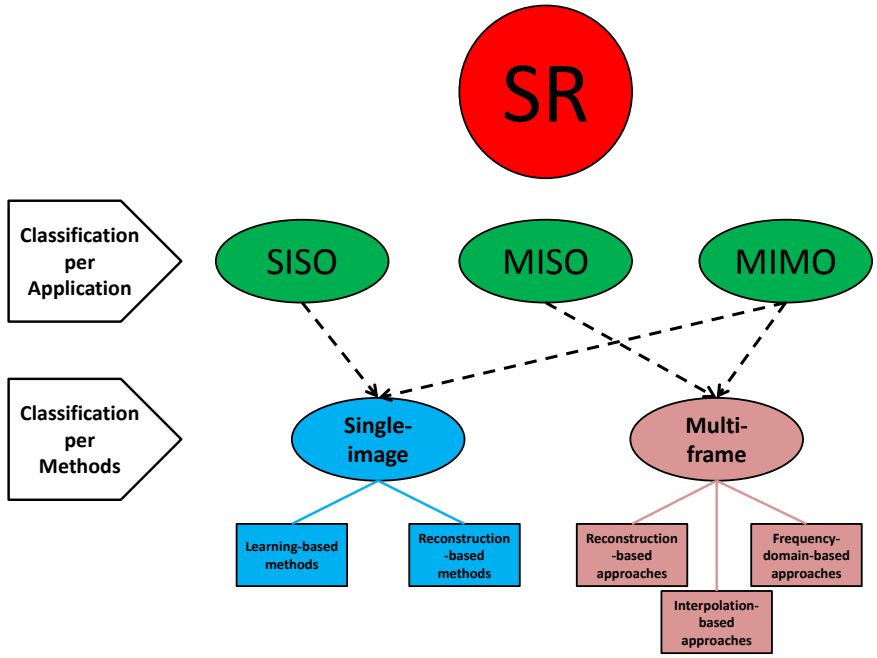


Figure A.1: Schéma taxonomique de la super-résolution - Classification des algorithmes de SR d'un point de vue des applications et des méthodologies réellement utilisées.

1. Une phase d'apprentissage, où le dictionnaire de blocs mentionné ci-dessus est construit, et

2. La phase de super-résolution propre, qui utilise le dictionnaire créé pour agrandir en HR l'image d'entrée.

À propos de la phase d'apprentissage, le dictionnaire peut être de deux types : externe, c'est-à-dire construit à partir d'un ensemble d'images d'apprentissage externes, ou interne, c'est-à-dire construit en utilisant seulement l'image d'entrée LR elle-même.

Quant à la phase de super-résolution, les algorithmes basés exemples consistent en des procédures opérant sur des blocs. En effet, l'image d'entrée BR est partitionnée en blocs. Puis, pour chaque bloc d'entrée BR, un bloc de sortie HR est construit, en utilisant les correspondances de blocs BR-HR dans le dictionnaire. L'image de sortie HR est enfin construite par réassemblage de tous les blocs HR construits.

Chapitre 2 : SR single-image basée sur la méthode "nonnegative neighbor embedding" et sur l'utilisation d'un dictionnaire externe

Dans les méthodes de SR basée exemples, il y a deux aspects discriminants : le type de dictionnaire utilisé et la méthode de reconstruction des blocs appliquée lors de la procédure de SR. Dans ce chapitre, nous concevons des algorithmes qui utilisent un dictionnaire externe et du "neighbor embedding" (NE) comme méthode de reconstruction des patches. Les algorithmes présentés sont le résultat de plusieurs contributions, selon une amélioration progressive d'une proposition initiale.

- A partir de la procédure générale de SR basée NE, décrite dans la Section 2.1, nous proposons une nouvelle procédure basée sur du NE non-negatif (Section 2.2). Nous appelons cette méthode *NoNNE* ("Non-Negative Neighbor Embedding").
- Une nouvelle méthode pour construire un dictionnaire plus compact et performant est ensuite présentée dans la Section 2.3. Nous appelons cette nouvelle procédure de construction du dictionnaire *JKC* ("Joint K-means Clustering").
- Enfin, un nouvel outil d'interpolation "améliorée" et de nouveaux schémas d'apprentissage pour extraire les blocs sont introduits dans la Section 2.4 L'algorithme, qui utilise ces nouveaux outils ainsi que la procédure JKC, est appelé *NEEB* ("Neighbor Embedding using Enhanced Bicubic interpolation").

Chapitre 3 : SR single-image basée sur des projections linéaires et l'utilisation d'un dictionnaire interne

Dans ce chapitre, nous présentons un nouvel algorithme de SR basée exemples, qui est en quelque sorte "symétrique" aux algorithmes présentés dans le Chapitre 2 . Différemment de ces derniers, qui sont basés sur l'approche NE, cet algorithme tombe dans la catégorie des méthodes de mappage direct (MD). Chaque bloc de sortie HR est le résultat d'une opération de projection linéaire avec une fonction précédemment apprise, appliquée directement sur le patch d'entrée BR correspondant.

Quant à l'autre aspect de discrimination des méthodes de SR basée exemples, la typologie du dictionnaire, nous explorons ici la possibilité d'avoir un diction-

naire interne, en ayant comme objectif la maximisation de la qualité de sortie. En particulier, le dictionnaire interne est construit par une "pyramide double", où la pyramide d'images traditionnelle de [58] est juxtaposée avec une pyramide d'images interpolées, et la procédure de reconstruction basée sur un MD consiste en l'apprentissage d'une fonction linéaire qui est appliquée sur les blocs interpolés.

Prenant comme référence l'algorithme bien connu de [58], les principales contributions sont alors :

- La modification du schéma d'apprentissage et de sur-échantillonnage (soit la pyramide double), et
- L'emploi d'une méthode MD dans les reconstructions, alors que dans [58] les blocs HR sont reconstruits par NE.

Chapitre 4 : Super-résolution d'une séquence vidéo

Dans le Chapitre 4 nous abordons le problème de l'extension de la théorie et des résultats développés dans les chapitres précédents de ce manuscrit (en particulier, sur les algorithmes de SR basée exemples qui utilisent soit un dictionnaire externe soit un dictionnaire interne) au cas de la vidéo.

Comme dans ce doctorat nous avons surtout mis l'accent sur la conception d'algorithmes appartenant à la catégorie de la SR single-image, nous voulons adapter ces algorithmes au cas de la SR vidéo-à-vidéo. Le problème de la SR vidéo est alors abordé du point de vue single-image où, fondamentalement, chaque image est agrandie séparément. Des questions telles que la cohérence temporelle entre les différentes images sont également prises en compte.

Lors de la conversion d'une séquence vidéo, nous pouvons envisager deux scénarios.

- *Scénario A* : La séquence vidéo contient aussi des images HR clés, apparaissant avec une fréquence fixe f_K (voir la Figure A.2).
- *Scénario B* : La séquence vidéo ne contient que des images BR (voir la Figure A.3)

Les deux problèmes de SR liés aux scénarios envisagés (Scénario A et Scénario B) sont abordés, respectivement, dans la Section 4.2 et la Section 4.3. Avant cela, dans la Section 4.1, nous introduisons une nouvelle méthode de calcul des poids de reconstruction dans le cas de la SR basée NE qui, en agrandissant une certaine image, prend également en compte les images précédemment reconstruitées. Ces poids sont destinés à améliorer la cohérence temporelle entre les images reconstruites avec une méthode de SR single-image, et peuvent peut être utilisés dans les deux scénarios.

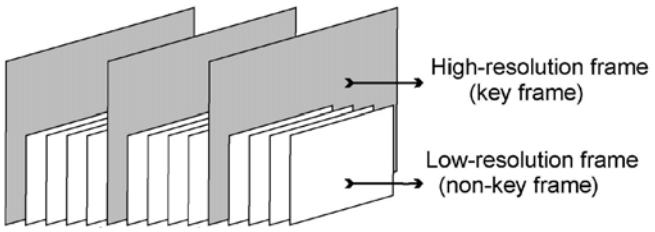


Figure A.2: ***Scénario A*** **consideré pour le sur-échantillonnage d'une séquence vidéo** - Hormis les images BR, des images HR périodiques apparaissent dans la séquence.

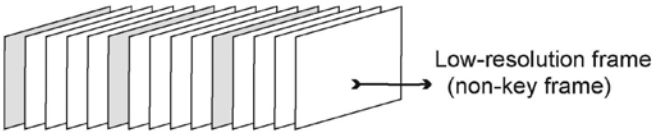


Figure A.3: ***Scénario B*** **consideré pour le sur-échantillonnage d'une séquence vidéo** - La séquence consiste uniquement d'images BR.

Chapitre 5 : Classification des techniques d'apprentissage utilisées dans les méthodes de SR proposées

Dans ce dernier chapitre, nous classons des techniques appartenant au domaine de l'apprentissage automatique, que nous avons étudiées et utilisées pour le développement d'algorithmes de SR. En particulier, après l'introduction dans la Section 5.1 de concepts basics sur l'apprentissage, dans la Section 5.2 nous présentons la régression comme une méthode d'apprentissage supervisé, en le rapportant à nos méthodes de SR basée exemples. Ensuite, dans la Section 5.3, nous fournissons un recueil de quelques techniques d'apprentissage non supervisé que nous avons particulièrement étudiées, notamment certaines méthodes pour effectuer l'apprentissage de dictionnaire et le partitionnement de données.

Dans le cadre de ces techniques, nous présentons enfin dans la Section 5.3.5 une nouvelle méthode que nous avons conçue, pour apprendre un dictionnaire d'atomes non négatifs particulièrement adapté pour les représentations parcimonieuses. Cette nouvelle méthode, que nous avons appelée *K-WEB* (car elle implique le calcul de "K WEighted Barycentres"), est évaluée en effectuant des comparaisons avec d'autres méthodes de l'état de l'art.

Chapitre 6 : Conclusions et perspectives

Dans cette thèse, nous avons examiné de manière approfondie le problème de la SR single-image, présentant différents algorithmes appartenant à la famille de la SR basée exemples. Plus précisément, nous avons conçu trois algorithmes de SR single-image, chacun présentant ses propres avantages :

- L'algorithme *NoNNE*, qui représente un bon compromis entre qualité et faible complexité,
- L'algorithme *NEEB*, qui utilise également une procédure de construction du dictionnaire *JKC*, et qui est une des méthodes les plus performantes parmi celles utilisant un dictionnaire externe, et
- L'algorithme "pyramide double", dont l'objectif est d'optimiser la qualité des images de sortie, en utilisant un dictionnaire interne.

En outre, dans le Chapitre 4, nous avons analysé le cas de la vidéo, c'est-à-dire le sur-échantillonnage d'une séquence vidéo entière, et fourni une analyse intéressante dans le contexte du codage vidéo. Enfin, dans le Chapitre 5, nous avons passé en révise toutes les techniques de SR proposées du point de vue de l'apprentissage automatique.

Ces dernières années, de nombreux algorithmes, qui montrent déjà des résultats très prometteurs, ont été développés pour le problème de SR single-image. Nous croyons donc que l'attention à ce sujet devrait maintenant basculer de la conception de nouvelles procédures à d'autres aspects importants, comme la validation théorique et la mise en œuvre pratique. À propos du premier aspect, les méthodes de SR, en particulier les méthodes basées exemples, sont souvent le résultat d'une approche heuristique. Nous pensons alors que certaines études supplémentaires peuvent être menées pour dévoiler complètement le lien entre la SR basée exemples et l'apprentissage automatique, comme nous l'avons fait en partie dans le Chapitre 5. Quant à la mise en œuvre pratique des méthodes de SR single-image, nous croyons qu'il y a de l'espace pour optimiser ces dernières, et donc les utiliser en tant que méthodes rapides de sur-échantillonnage de la même manière que les méthodes d'interpolation classiques.

Enfin, nous pensons qu'il y a encore quelques applications où la SR pourrait être utilisée. Un exemple est donné par les images plénoptiques, qui peuvent être prises par certains appareils photographiques récemment commercialisés et qui consistent en une série d'images BR se référant à différents points de vue.

Glossary

ARPS	Adaptive rood pattern search
CFT	Continuous Fourier transform
CIF	Common intermediate format
CLS	Constrained least squares
DCT	Discrete cosine transform
DFT	Discrete Fourier transform
DM	Direct mapping
EM	Expectation maximization
GOP	Group of pictures
HF	High-frequency
HR	High-resolution
IBP	Iterative back-projection
JKC	Joint k-means clustering
K-SVD	K-singular value decomposition
K-WEB	K weighted barycenters
LLE	Locally linear embedding
LR	Low-resolution
LS	Least squares
MAP	Maximum a posterior
MEA	Motion-estimation-aided
MIMO	Multiple-input multiple-output
MISO	Multiple-input single-output

ML	Maximum likelihood
MLR	Multilinear regression
MV	Motion vector
NE	Neighbor embedding
NEEB	Neighbor Embedding SR using Enhanced Bicubic interpolation
NMF	Nonnegative matrix factorization
NMP	Nonnegative matching pursuit
NNBP	Nonnegative basis pursuit
NNS	Nearest neighbor search
NoNNE	NonNegative Neighbor Embedding
OMP	Orthogonal matching pursuit
PDF	Probably density function
POCS	Projection onto convex sets
QCIF	Quarter CIF
QP	Quantization parameter
SISO	Single-input single-output
SNMF	Semi-nonnegative matrix factorization
SR	Super-resolution
SVD	Singular value decomposition
TV	Total variation
VQ	Vector quantization
WCSS	Within-cluster sum of squares

Bibliography

[1] H. Greenspan, "Super-Resolution in Medical Imaging," *The Computer Journal*, vol. 52, no. 1, pp. 43–63, Jan. 2009.

[2] M. T. Merino and J. Nunez, "Super-Resolution of Remotely Sensed Images With Variable-Pixel Linear Reconstruction," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 5, pp. 1446–1457, May 2007.

[3] A. J. Tatem, H. G. Lewis, P. M. Atkinson, and M. S. Nixon, "Super-resolution target identification from remotely sensed images using a Hopfield neural network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, no. 4, pp. 781–796, Apr. 2001.

[4] E. Engin and M. Özcan, "Moving target detection using super-resolution algorithms with an ultra wideband radar," *International Journal of Imaging Systems and Technology*, vol. 20, no. 3, pp. 237–244, 2010.

[5] S. Ebihara, M. Sato, and H. Niitsuma, "Super-Resolution of Coherent Targets by a Directional Borehole Radar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 4, pp. 1725–1732, Jul. 2000.

[6] A. Gehani and J. H. Reif, "Super-Resolution Video Analysis for Forensic Investigations," in *IFIP WG 11.9 International Conference on Digital Forensics*, ser. IFIP, vol. 242. Springer, 2007, pp. 281–299.

[7] Y. Wang, R. Fevig, and R. R. Schultz, "Super-resolution mosaicking of UAV surveillance video," in *IEEE International Conference on Image Processing (ICIP)*, 2008, pp. 345–348.

[8] T. S. Huang and R. Y. Tsai, "Multiframe image restoration and registration," *Advances in Computer Vision and Image Processing*, vol. 1, no. 7, pp. 317–339, 1984.

[9] M. Irani and S. Peleg, "Super resolution from image sequences," in *10th International Conference on Pattern Recognition*, vol. 2, Jun. 1990, pp. 115–120.

[10] S. Borman and R. Stevenson, "Super-Resolution from Image Sequences – A Review," in *Midwest Symposium on Circuits and Systems*, Notre Dame, IN, USA, 8 1998, pp. 374–378.

[11] S. C. Park, M. K. Park, and M. G. Kang, "Super-Resolution Image Reconstruction: A Technical Overview," *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 21–36, 5 2003.

[12] C. Mancas-Thillou and M. Mirmehdi, "An Introduction to Super-Resolution Text," in *Digital Document Processing*, ser. Advances in Pattern Recognition. Springer London, 2007, pp. 305–327.

[13] J. Tian and K.-K. Ma, "A survey on super-resolution imaging," *Signal, Image and Video Processing (SIViP)*, vol. 5, no. 3, pp. 329–342, 2011.

[14] H. Ur and D. Gross, "Improved Resolution from Subpixel Shifted Pictures," *CVGIP: Graph. Models Image Process.*, vol. 54, no. 2, pp. 181–186, Mar. 1992.

[15] A. Papoulis, "Generalized Sampling Expansion," *IEEE Transactions on Circuits and Systems,*, vol. 24, no. 11, pp. 652–654, Nov. 1977.

[16] N. K. Bose and N. A. Ahuja, "Superresolution and Noise Filtering Using Moving Least Squares," *IEEE Transactions on Image Processing*, vol. 15, no. 8, pp. 2239–2248, Aug. 2006.

[17] A. J. Patti, M. I. Sezan, and A. M. Tekalp, "Superresolution Video Reconstruction with Arbitrary Sampling Lattices and Nonzero Aperture Time," *IEEE Transactions on Image Processing*, vol. 6, no. 8, pp. 1064–1076, Aug. 1997.

[18] S. Rhee and M. G. Kang, "DCT-Based Regularized Algorithm for High-Resolution Image Reconstruction," in *IEEE International Conference on Image Processing (ICIP)*. Los Alamitos, CA: IEEE, Oct. 1999, pp. 184–187.

[19] N. A. Woods, N. P. Galatsanos, and A. K. Katsaggelos, "Stochastic Methods for Joint Registration, Restoration, and Interpolation of Multiple Undersampled Images," *IEEE Transactions on Image Processing*, vol. 15, no. 1, pp. 201–213, Jan. 2006.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[21] N. Nguyen and P. Milanfar, "An Efficient Wavelet-Based Algorithm for Image Superresolution," in *IEEE Internation Conference on Image Processing*, 2000, pp. 351–354.

[22] S. E. El-Khamy, M. M. Hadhoud, M. I. Dessouky, B. M. Salam, and F. E. A. El-Samie, "Wavelet Fusion: a Tool to Break the Limits on LMMSE Image Super-Resolution," *International Journal of Wavelets, Multiresolution and Information Processing (IJWMIP)*, vol. 4, no. 1, pp. 105–118, 2006.

[23] H. Ji and C. Fermüller, "Wavelet-Based Super-Resolution Reconstruction: Theory and Al-

gorithm," in *9th European Conference on Computer Vision (ECCV)*, vol. 3954. Springer, 2006, pp. 295–307.

[24] ——, "Robust Wavelet-Based Super-Resolution Reconstruction: Theory and Algorithm," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, Apr. 2009, pp. 649–660.

[25] M. B. Chappalli and N. K. Bose, "Simultaneous Noise Filtering and Super-Resolution With Second-Generation Wavelets," *IEEE Signal Processing Letters*, vol. 12, no. 11, pp. 772–775, Nov. 2005.

[26] A. K. Katsaggelos, *Digital Image Restoration*. Heidelberg, Germany: Springer, 1991, vol. 23.

[27] R. C. Hardie, K. J. Barnard, and E. E. Armstrong, "Joint MAP registration and high-resolution image estimation using a sequence of undersampled images," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1621–1633, 1997.

[28] J. Tian and K.-K. Ma, "Stochastic super-resolution image reconstruction," *Jorunal of Visual Communication and Image Representation*, vol. 21, no. 3, pp. 232–244, 2010.

[29] R. R. Schultz and R. L. Stevenson, "Extraction of High-Resolution Frames from Video Sequences," *IEEE Transactions on Image Processing*, vol. 5, no. 6, pp. 996–1011, Jun. 1996.

[30] K. V. Suresh and A. N. Rajagopalan, "Robust and computationally efficient superresolution algorithm," *Journal of the Optical Society of America*, vol. 24, no. 4, pp. 984–992, Apr. 2007.

[31] S. P. Belekos, N. P. Galatsanos, and A. K. Katsaggelos, "Maximum a Posteriori Video Super-Resolution Using a New Multichannel Image Prior," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1451–1464, 2010.

[32] H. Shen, L. Zhang, B. Huang, and P. Li, "A MAP Approach for Joint Motion Estimation, Segmentation, and Super Resolution," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 479–490, Feb. 2007.

[33] B. C. Tom and A. K. Katsaggelos, "Reconstruction of a high-resolution image by simultaneous registration, restoration, and interpolation of low-resolution images," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 1995, pp. 539–542.

[34] S. Baker and T. Kanade, "Limits on Super-Resolution and How to Break Them," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1167–1183, 2002.

[35] R. G. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, no. 6, pp. 1153–1160, Dec. 1981.

[36] X. Li and M. T. Orchard, "New Edge-Directed Interpolation," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1521–1527, Oct. 2001.

[37] M. F. Tappen, B. C. Russell, and W. T. Freeman, "Exploiting the Sparse Derivative Prior for Super-Resolution and Image Demosaicing," in *3rd IEEE International Workshop on Statistical and Computational Theories of Vision*, 2003.

[38] S. Dai, M. Han, W. Xu, Y. Wu, and Y. Gong, "Soft Edge Smoothness Prior for Alpha Channel Super Resolution," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2007, pp. 1–8.

[39] S. Dai, M. Han, W. Xu, Y. Wu, Y. Gong, and A. Katsaggelos, "SoftCuts: A Soft Edge Smoothness Prior for Color Image Super-Resolution," *IEEE Transactions on Image Processing*, vol. 18, no. 5, pp. 969–981, 5 2009.

[40] R. Fattal, "Image Upsampling via Imposed Edge Statistics," *ACM Transactions on Graphics*, vol. 26, no. 3, pp. 1–8, Jul. 2007.

[41] H. Aly and E. Dubois, "Image up-sampling using total-variation regularization with a new observation model," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1647–1659, Oct. 2005.

[42] P. Sen and S. Darabi, "Compressive Image Super-resolution," in *43rd Asilomar Conference on Signals, Systems and Computers*, 11 2009, pp. 1235–1242.

[43] W. Dong, D. Zhang, G. Shi, and X. Wu, "Image Deblurring and Super-Resolution by Adaptive Sparse Domain Selection and Adaptive Regularization," *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 1838–1857, Jul. 2011.

[44] W. Dong, L. Zhang, R. Lukac, and G. Shi, "Sparse Representation based Image Interpolation with Nonlocal Autoregressive Modeling," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1382–1394, Apr. 2013.

[45] Q. Shan, Z. Li, J. Jia, and C.-K. Tang, "Fast Image/Video Upsampling," *ACM Transactions on Graphics*, vol. 27, no. 5, pp. 153:1–153:7, Dec. 2008.

[46] H. He and W.-C. Siu, "Single image super-resolution using Gaussian process regression," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 449–456.

[47] K. Zhang, X. Gao, D. Tao, and X. Li, "Single Image Super-Resolution With Non-Local

Means and Steering Kernel Regression," *IEEE Transactions on Image Processing*, vol. 21, no. 11, pp. 4544–4556, 2012.

[48] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-Based Super-Resolution," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.

[49] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-Resolution Through Neighbor Embedding," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2004, pp. 275–282.

[50] W. Fan and D.-Y. Yeung, "Image Hallucination Using Neighbor Embedding over Visual Primitive Manifolds," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 7 2007, pp. 1–7.

[51] T.-M. Chan, J. Zhang, J. Pu, and H. Huang, "Neighbor embedding based super-resolution algorithm through edge detection and feature selection," *Pattern Recognition Letters*, vol. 30, no. 5, pp. 494–502, 4 2009.

[52] J. Yang, J. Wright, T. Huang, and Y. Ma, "Image Super-Resolution Via Sparse Representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861 –2873, 11 2010.

[53] J. Wang, S. Zhu, and Y. Gong, "Resolution enhancement based on learning the sparse association of image patches," *Pattern Recognition Letters*, vol. 31, pp. 1–10, 1 2010.

[54] R. Zeyde, M. Elad, and M. Protter, "On Single Image Scale-Up Using Sparse-Representations," in *Curves and Surfaces*, ser. Lecture Notes in Computer Science, J.-D. Boissonnat, P. Chenin, A. Cohen, C. Gout, T. Lyche, M.-L. Mazure, and L. Schumaker, Eds. Springer Berlin Heidelberg, 2012, vol. 6920, pp. 711–730.

[55] X. Gao, K. Zhang, D. Tao, and X. Li, "Image Super-Resolution With Sparse Neighbor Embedding," *IEEE Transactions on Image Processing*, vol. 21, no. 7, pp. 3194–3205, 2012.

[56] G. Freedman and R. Fattal, "Image and Video Upscaling from Local Self-Examples," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 1–10, 2010.

[57] J. Yang, Z. Lin, and S. Cohen, "Fast Image Super-Resolution Based on In-Place Example Regression," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2013, pp. 1059–1066.

[58] D. Glasner, S. Bagon, and M. Irani, "Super-Resolution from a Single Image," in *IEEE 12th International Conference on Computer Vision (ICCV)*, 10 2009, pp. 349–356.

[59] C.-Y. Yang, J.-B. Huang, and M.-H. Yang, "Exploiting Self-Similarities for Single Frame Super-Resolution," in *Asian Conference on Computer Vision (ACCV)*, Nov. 2010.

[60] M.-C. Yang, C.-H. Wang, T.-Y. Hu, and Y.-C. F. Wang, "Learning context-aware sparse representation for single image super-resolution," in *IEEE International Conference on Image Processing (ICIP)*, Sep. 2011, pp. 1349–1352.

[61] M. Zontak and M. Irani, "Internal Statistics of a Single Natural Image," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 977–984.

[62] A. Buades, B. Coll, and J.-M. Morel, "A Non-Local Algorithm for Image Denoising," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, Washington, DC, USA, 2005, pp. 60–65.

[63] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 12 2000.

[64] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. A. Morel, "Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding," in *Proceedings of the British Machine Vision Conference (BMVC)*. Guildford, UK: BMVA Press, Sep. 2012, pp. 135.1–135.10.

[65] Y. Tang, P. Yan, Y. Yuan, and X. Li, "Single-image super-resolution via local learning," *International Journal of Machine Learning and Cybernetics*, vol. 2, pp. 15–23, 2011.

[66] K. I. Kim and Y. Kwon, "Single-Image Super-Resolution Using Sparse Regression and Natural Image Prior," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 6, pp. 1127–1133, 2010.

[67] J. Yang, J. Wright, T. Huang, and Y. Ma, "Image Super-Resolution as Sparse Representation of Raw Image Patches," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2008, pp. 1–8.

[68] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[69] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient Sparse Coding Algorithms," in *20th Annual Conference on Neural Information Processing Systems*, vol. 19, 2006, pp. 801–808.

[70] L. K. Saul and S. T. Roweis, "Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds," *Journal of Machine Learning Research*, vol. 4, pp. 119–155, 12 2003.

[71] K. Su, Q. Tian, Q. Xue, N. Sebe, and J. Ma, "Neighborhood issue in Single-frame Image Super-Resolution," in *IEEE International Conference on Multimedia and Expo (ICME)*, 7 2005, pp. 1122–1125.

[72] J. Sun, N. ning Zheng, H. Tao, and H. yeung Shum, "Image hallucination with primal sketch priors," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003, pp. 729–736.

[73] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. Alberi Morel, "Neighbor embedding based single-image super-resolution using Semi-Nonnegative Matrix Factorization," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, Mar. 2012, pp. 1289–1292.

[74] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 10 1999.

[75] R. J. H. Charles L. Lawson, *Solving Least Squares Problems.* Prentice-Hall, 1974.

[76] C. Ding, T. Li, and M. I. Jordan, "Convex and Semi-Nonnegative Matrix Factorizations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 45–55, 11 2010.

[77] S.-C. Jeong and B. C. Song, "Fast Super-Resolution Algorithm Based on Dictionary Size Reduction Using k-Means Clustering," *ETRI Journal*, vol. 32, no. 4, pp. 596–602, 8 2010.

[78] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. Alberi Morel, "Compact and coherent dictionary construction for example-based super-resolution," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

[79] B. Li, H. Chang, S. Shan, and X. Chen, "Locality preserving constraints for super-resolution with neighbor embedding," in *IEEE International Conference on Image Processing (ICIP)*, 11 2009, pp. 1189–1192.

[80] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982.

[81] D. Arthur and S. Vassilvitskii, "K-means++: The Advantages of Careful Seeding," in *18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007, pp. 1027–1035.

[82] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. Alberi Morel, "Super-resolution using Neighbor Embedding of Back-projection residuals," in *18th International Conference on Digital Signal Processing (DSP)*, Santorini, Greece, Jul. 2013.

[83] H. Hyötyniemi, "Multivariate regression - Techniques and tools," Helsinki University of Technology, Control Engineering Laboratory, Helsinki, Finland, Tech. Rep. Report 125, 2001.

[84] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. Alberi Morel, "On Dictionary Learning

for Example-Based Super-Resolution," *Bell Labs Technical Journal*, vol. 19, no. 1, pp. 1–22, Jun. 2014, accepted pending revisions.

[85] ——, "Video super-resolution via sparse combinations of key-frame patches in a compression context," in *30th Picture Coding Symposium (PCS)*, San Jose, CA, USA, Dec. 2013.

[86] E. Hung, R. De Queiroz, F. Brandi, K. de Oliveira, and D. Mukherjee, "Video Super-Resolution Using Codebooks Derived From Key-Frames," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 9, pp. 1321–1331, 2012.

[87] Y. Nie and K.-K. Ma, "Adaptive rood pattern search for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 11, no. 12, pp. 1442–1449, 2002.

[88] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[89] M. Aharon, M. Elad, and A. M. Bruckstein, "K-SVD and its non-negative variant for dictionary design," in *Proceedings of the SPIE conference wavelets*, 2005, pp. 327–339.

[90] D. D. Lee and H. S. Seung, "Algorithms for Non-negative Matrix Factorization," in *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*. MIT Press, 4 2001, pp. 556–562.

[91] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. Alberi Morel, "K-WEB: Nonnegative dictionary learning for sparse image representations," in *IEEE International Conference on Image Processing (ICIP)*, Melbourne, Australia, Sep. 2013.

[92] R. Peharz, M. Stark, and F. Pernkopf, "Sparse nonnegative matrix factorization using ℓ^0-constraints," in *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 8 2010, pp. 83–88.

[93] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal Matching Pursuit : recursive function approximation with application to wavelet decomposition," in *Asilomar Conf. on Signals, Systems and Computer*, 1993.

List of Figures

List of Tables

Abstract

With super-resolution (SR) we refer to a class of techniques that enhance the spatial resolution of images and videos. SR algorithms can be of two kinds: multi-frame methods, where multiple low-resolution images are aggregated to form a unique high-resolution image, and single-image methods, that aim at upscaling a single image. This thesis focuses on developing theory and algorithms for the single-image SR problem. In particular, we adopt the so called example-based approach, where the output image is estimated with machine learning techniques, by using the information contained in a dictionary of image "examples". The examples consist in image patches, which are either extracted from external images or derived from the input image itself. For both kinds of dictionary, we design novel SR algorithms, with new upscaling and dictionary construction procedures, and compare them to state-of-the-art methods. The results achieved are shown to be very competitive both in terms of visual quality of the super-resolved images and computational complexity. We then apply our designed algorithms to the video upscaling case, where the goal is to enlarge the resolution of an entire video sequence. The algorithms, opportunely adapted to deal with this case, are also analyzed in the coding context. The analysis conducted shows that, in specific cases, SR can also be an effective tool for video compression, thus opening new interesting perspectives.

Résumé

Par le terme "super-résolution" (SR), nous faisons référence à une classe de techniques qui améliorent la résolution spatiale d'images et de vidéos. Les algorithmes de SR peuvent être de deux types : les méthodes "multi-frame", où plusieurs images en basse résolution sont agrégées pour former une image unique en haute résolution, et les méthodes "single-image", qui visent à élargir une seule image. Cette thèse a pour sujet le développement de théories et algorithmes pour le problème single-image. En particulier, nous adoptons une approche "basée sur exemples", où l'image de sortie est estimée grâce à des techniques d'apprentissage automatique, en utilisant les informations contenues dans un dictionnaire d'exemples. Ces exemples consistent en des blocs d'image, soit extraits à partir d'images externes, soit dérivées de l'image d'entrée elle-même. Pour les deux types de dictionnaire, nous concevons de nouveaux algorithmes de SR présentant de nouvelles méthodes de suréchantillonnage et de construction du dictionnaire, et les comparons à l'état de l'art. Les résultats obtenus s'avèrent très compétitifs en termes de qualité visuelle des images de sortie et de complexité des calculs. Nous appliquons ensuite nos algorithmes au cas de la vidéo, où l'objectif est d'élargir la résolution d'une séquence vidéo. Les algorithmes, opportunément adaptées pour faire face à ce cas, sont également analysés dans le contexte du codage. L'analyse effectuée montre que, dans des cas spécifiques, la SR peut aussi être un outil efficace pour la compression vidéo, ouvrant ainsi de nouvelles perspectives intéressantes.