On Single Image Scale-Up using Sparse-Representations

Roman Zeyde, Michael Elad and Matan Protter

The Computer Science Department
Technion – Israel Institute of Technology – Haifa 32000, Israel
{romanz,elad,matanpr}@cs.technion.ac.il

Abstract. This paper deals with the single image scale-up problem using sparse-representation modeling. The goal is to recover an original image from its blurred and down-scaled noisy version. Since this problem is highly ill-posed, a prior is needed in order to regularize it. The literature offers various ways to address this problem, ranging from simple linear space-invariant interpolation schemes (e.g., bicubic interpolation), to spatially-adaptive and non-linear filters of various sorts.
We embark from a recently-proposed successful algorithm by Yang et. al. [13,14], and similarly assume a local *Sparse-Land* model on image patches, serving as regularization. Several important modifications to the above-mentioned solution are introduced, and are shown to lead to improved results. These modifications include a major simplification of the overall process both in terms of the computational complexity and the algorithm architecture, using a different training approach for the dictionary-pair, and introducing the ability to operate without a training-set by boot-strapping the scale-up task from the given low-resolution image. We demonstrate the results on true images, showing both visual and PSNR improvements.

1 Introduction

Many applications require resolution enhancement of images acquired by low-resolution sensors (e.g. for high-resolution displays), while minimizing visual artifacts. The single image scale-up[1] problem can be formulated as follows: denote the original high-resolution image as $\mathbf{y}_h \in \mathbb{R}^{N_h}$, represented as a vector of length N_h pixels. In addition, denote the blur and decimation operators as $\mathbf{H} : \mathbb{R}^{N_h} \to \mathbb{R}^{N_h}$ and $\mathbf{S} : \mathbb{R}^{N_h} \to \mathbb{R}^{N_l}$ (where $N_l < N_h$) respectively. It is assumed hereafter that $\mathbf{H}$ applies a known low-pass filter to the image, and $\mathbf{S}$ performs a decimation by an integer factor s, by discarding rows/columns from the input image. $\mathbf{z}_l \in \mathbb{R}^{N_l}$ is defined to be the low-resolution noisy version of the original image as

$$\mathbf{z}_l = \mathbf{SH}\mathbf{y}_h + \mathbf{v}, \tag{1}$$

[1] This problem is often referred to in the literature as *super-resolution*. The term *super-resolution* may be confusing, as it is also used in the context of fusing several low-resolution images into one high-resolution result.[9]

for an additive i.i.d. white Gaussian noise, denoted by $\mathbf{v} \sim \mathcal{N}\left(0, \sigma^2 \mathbf{I}\right)$.

Given $\mathbf{z}_l$, the problem is to find $\hat{\mathbf{y}} \in \mathbb{R}^{N_h}$ such that $\hat{\mathbf{y}} \approx \mathbf{y}_h$. Due to the Gaussianity of $\mathbf{v}$, the maximum-likelihood estimation is obtained by the minimization of $\|\mathbf{SH}\hat{\mathbf{y}} - \mathbf{z}_l\|_2$. However, since $\mathbf{SH}$ is rectangular with more columns than rows, it cannot be inverted stably, and there are infinitely many solutions that lead to a zero value in the above-mentioned least-squares term. Existing single-image scale-up algorithms use various priors on the image in order to stabilize this inversion, ranging from Tikhonov regularization, robust statistics and Total-Variation, sparsity of transform coefficients, and all the way to example-based techniques that use training set of images as priors. While we do not provide a comprehensive review of these techniques, we refer the reader to the following papers [1,2,3,4,5,6,7,8].

In this work we shall use the *Sparse-Land* local model, as introduced in [10,11,12], for the scale-up problem. This model assumes that each patch from the images considered can be well represented using a linear combination of few atoms from a dictionary. Put differently, each patch is considered to be generated by multiplying a dictionary by a sparse (mostly zero) vector of coefficients. This assumption will help us in developing an algorithm for image scale-up. It is important to note that this is also the path taken by [13,14] and similar to [18]. However, our work differs from their solution in several important aspects, as described in the paper.

This paper is organized as follows: The incorporation of the *Sparse-Land* model into the scale-up problem is shown in Section 2. Section 3 describes the actual implementation details of the algorithm. Experiments and comparative results are given in Section 4, and conclusions are drawn in Section 5.

2 Incorporating the Sparse-Land Prior

In order to avoid the complexities caused by the different resolutions between $\mathbf{z}_l$ and $\mathbf{y}_h$, and in order to simplify the overall recovery algorithm, it is assumed hereafter that the image $\mathbf{z}_l$ is scaled-up by a simple interpolation operator $\mathbf{Q} : \mathbb{R}^{N_l} \rightarrow \mathbb{R}^{N_h}$ (e.g. bicubic interpolation) that fills-in the missing rows/columns, returning to the size of $\mathbf{y}_h$. This decision will not badly influence the computational complexity of the algorithm, and in fact, the eventual scale-up algorithm proposed here is much faster than the one proposed in [13,14,18]. The scaled-up image shall be denoted by $\mathbf{y}_l$ and it satisfies the relation

$$\mathbf{y}_l = \mathbf{Q}\mathbf{z}_l = \mathbf{Q}\left(\mathbf{SH}\mathbf{y}_h + \mathbf{v}\right) = \mathbf{QSH}\mathbf{y}_h + \mathbf{Q}\mathbf{v} = \mathbf{L}^{all}\mathbf{y}_h + \tilde{\mathbf{v}}. \tag{2}$$

The goal is to process $\mathbf{y}_l \in \mathbb{R}^{N_h}$ and produce a result $\hat{\mathbf{y}}_h \in \mathbb{R}^{N_h}$, which will get as close as possible to the original high-resolution image, $\mathbf{y}_h \in \mathbb{R}^{N_h}$.

The algorithm we propose operates on patches extracted from $\mathbf{y}_l$, aiming to estimate the corresponding patch from $\mathbf{y}_h$. Let $\mathbf{p}_h^k = \mathbf{R}_k\mathbf{y}_h \in \mathbb{R}^n$ be a high-resolution image patch of size $\sqrt{n} \times \sqrt{n}$, extracted by the operator $\mathbf{R}_k : \mathbb{R}^{N_h} \rightarrow \mathbb{R}^n$ from the image $\mathbf{y}_h$ in location k. It is assumed that the locations to consider $\{k\}$ are only those centered around true pixels in the low-resolution image $\mathbf{y}_l$

(as opposed to filled-in pixels due to the interpolation). This set of samples is referred to hereafter as the set Ω.

It is now time to invoke the *Sparse-Land* model: it shall be further assumed that $\mathbf{p}_h^k \in \mathbb{R}^n$ can be represented sparsely by $\mathbf{q}^k \in \mathbb{R}^m$ over the dictionary $\mathbf{A}_h \in \mathbb{R}^{n \times m}$, namely:

$$\mathbf{p}_h^k = \mathbf{A}_h \mathbf{q}^k, \tag{3}$$

where $\|\mathbf{q}^k\|_0 \ll n$, where the ℓ_0-pseudo-norm counts the number of non-zeros in the vector $\mathbf{q}^k$. The matrix A_h is the dictionary that characterizes the high-resolution patches. Its construction will be discussed in details in Section 3.

Consider the corresponding low-resolution patch $\mathbf{p}_l^k = \mathbf{R}_k \mathbf{y}_l$, extracted from $\mathbf{y}_l$ in the same location (the patches $\mathbf{p}_l^k$ and $\mathbf{p}_h^k$ are centered around the same pixel k), such that its size is $\sqrt{n} \times \sqrt{n}$. Since the operator $\mathbf{L}^{all} = \mathbf{QSH}$ transforms the complete high-resolution image $\mathbf{y}_h$ to the low-resolution one, $\mathbf{y}_l$, it can be assumed that $\mathbf{p}_l^k = \mathbf{L}\mathbf{p}_h^k + \tilde{\mathbf{v}}_k$, where $\mathbf{L}$ is a local operator being a portion of $\mathbf{L}^{all}$, and $\tilde{\mathbf{v}}_k$ is the additive noise in this patch. Note that $\mathbf{L}$ is a spatially independent operator, as only locations $k \in \Omega$ from $\mathbf{y}_l$ are considered. [2]

Since it is assumed that $\mathbf{p}_h^k = \mathbf{A}_h \mathbf{q}^k$, multiplication of this equation by $\mathbf{L}$ gives

$$\mathbf{L}\mathbf{p}_h^k = \mathbf{L}\mathbf{A}_h \mathbf{q}^k. \tag{4}$$

Exploiting the relation between the low-resolution and the high-resolution patches $\mathbf{p}_l^k = \mathbf{L}\mathbf{p}_h^k + \tilde{\mathbf{v}}_k$, we obtain

$$\mathbf{L}\mathbf{A}_h \mathbf{q}^k = \mathbf{L}\mathbf{p}_h^k = \mathbf{p}_l^k - \tilde{\mathbf{v}}_k, \tag{5}$$

implying that

$$\|\mathbf{p}_l^k - \mathbf{L}\mathbf{A}_h \mathbf{q}^k\|_2 \leq \epsilon, \tag{6}$$

where ϵ is related to the noise power σ of $\mathbf{v}$.

The key observation from the above derivations is that the low-resolution patch $\mathbf{p}_l^k$ can be represented by the same sparse vector $\mathbf{q}^k$ over the effective dictionary $\mathbf{A}_l = \mathbf{L}\mathbf{A}_h$, with a controlled error ϵ. This implies that for a given low-resolution patch $\mathbf{p}_l^k$, its sparse representation vector, $\mathbf{q}^k$, is found and then $\mathbf{p}_h^k$ can be recovered by simply multiplying this representation by the dictionary $\mathbf{A}_h$. This is the core idea behind the image scale-up algorithm as developed by Yang et. al, [13,14], and we follow it as well, with important modifications.

[2] The observant reader might be troubled by boundary issues because of the spatial extent of the operator $\mathbf{L}_{all}$, and the fact that we have chosen the low-resolution and the high-resolution patches to be of the same size. However, the developed algorithm will not make use of the operator $\mathbf{L}$, and bypass this issue - more on this matter is brought in the next Section.

3 The Proposed Single-Image Scale-Up Algorithm

The scale-up algorithm consists of a training phase (that can be done off-line) as described in Figure 1 and a reconstruction phase, performing the scale-up on the test image using the trained model from the previous phase, as described in Figure 2.

1. Training set construction: A set of high-resolution training images $\{\mathbf{y}_h^j\}_j$ is collected, Low-resolution images $\{\mathbf{y}_l^j\}_j$ are constructed using scale-down operator $\mathbf{L}_{all}$ and pairs of matching patches that form the training database, $\mathcal{P} = \{\mathbf{p}_h^k, \mathbf{p}_l^k\}_k$, are extracted.
2. Each of these patch-pairs undergoes a pre-processing stage that removes the low-frequencies from $\mathbf{p}_h^k$ and extracts features from $\mathbf{p}_l^k$.
3. Dimensionality reduction is applied on the features of the low-resolution patches $\mathbf{p}_l^k$, making the dictionary training step much faster.
4. A dictionary $\mathbf{A}_l$ is trained for the low-resolution patches, such that they can be represented sparsely.
5. A corresponding dictionary $\mathbf{A}_h$ is constructed for the high-resolution patches, such that it matches the low-resolution one.

Fig. 1. Proposed algorithm's summary: training phase.

1. Given a test low-resolution image $\mathbf{z}_l$ to be scaled-up, it is interpolated to $\mathbf{y}_l$ of the destination size, and all that it requires is a spatial non-linear filtering to sharpen it.
2. Pre-processed patches $\mathbf{p}_l^k$ are extracted from each location $k \in \Omega$, and then sparse-coded using the trained dictionary $\mathbf{A}_l$.
3. The found representations $\{\mathbf{q}^k\}$ are then used to recover the high-resolution patches by multiplying them with $\mathbf{A}_h$.
4. The recovered high-resolution patches $\{\mathbf{p}_h^k\}$ are finally merged by averaging in the overlap area to create the resulting image.

Fig. 2. Proposed algorithm's summary: reconstruction phase.

3.1 Training Set Construction

The training phase starts by collecting several images $\{\mathbf{y}_h^j\}_j$, which are considered to be the high-resolution examples. Each of these images is blurred and

down-scaled by a factor of s. This leads to the formation of the corresponding low-resolution images $\{\mathbf{z}_l^j\}_j$, which are then scaled up back to the original size using $\mathbf{Q}$, resulting with the set $\{\mathbf{y}_l^j\}_j$. Thus,

$$\mathbf{y}_l^j = \mathbf{L}_{all}\mathbf{y}_h^j + \tilde{\mathbf{v}}^j. \tag{7}$$

It is important to note that the same operators $\mathbf{S}$, $\mathbf{H}$ and $\mathbf{Q}$ should be used both in the training and the reconstruction phases.

3.2 Preprocessing and Feature Extraction

The next step is pre-processing by high-pass filtering, similar to the approach in [4,13,14,18]. Rather than extracting small image-patches and applying this step on them, the desired pre-processing is employed directly on the full images, and only then are the patches extracted. This order avoids boundary problems due to the small patch size.[3]

The pre-processing applied to the high-resolution images consists of a removal of their low-frequencies, which is done by computing the difference images $\mathbf{e}_h^j = \mathbf{y}_h^j - \mathbf{y}_l^j$. The reason for this step is the desire to focus the training on characterizing the relation between the low-resolution patches and the edges and texture content within the corresponding high-resolution ones.

As for the pre-processing of the low-resolution images, these are filtered using R high-pass filters, in order to extract local features that correspond to their high-frequency content. Thus, each low-resolution image $\mathbf{y}_l^j$ results in a set of R filtered images, $\{f_r * \mathbf{y}_l^j\}_r$ for $r = 1, 2, \ \ldots\ , R$ (where $*$ stands for a convolution). Typical filters to be used may be gradient and Laplacian high-pass filters.

After the two pre-processing steps described above, local patches are extracted forming the data-set $\mathcal{P} = \{\mathbf{p}_h^k, \mathbf{p}_l^k\}_k$. Considering only locations $k \in \Omega$, $\mathbf{p}_h^k$ patches of size $\sqrt{n} \times \sqrt{n}$ pixels are extracted from the high-resolution images $\mathbf{e}_h^j$. The corresponding low-resolution $\mathbf{p}_l^k$ patches are extracted from the same locations in the filtered images $f_k * \mathbf{y}_l^j$ and using the same size ($\sqrt{n} \times \sqrt{n}$ pixels). Thus, every corresponding R such low-resolution patches are concatenated into one vector $\tilde{\mathbf{p}}_l^k$ of length nR. Note that the high-resolution patch size should be at least of size $s \times s$ so as to cover the high-resolution image. A larger patch-size results in overlaps between patches, which improves the reconstruction result (by reducing errors and discontinuities between reconstructed patches).

3.3 Dimensionality Reduction

The formed low-resolution patches, started as $\sqrt{n}/s \times \sqrt{n}/s = n/s^2$ pixel patches (in the images $\mathbf{z}_l^j$), are now represented as $\tilde{\mathbf{p}}_l^k$ of nR dimensions after an interpolation operator $\mathbf{Q}$ and set of R linear filters. As a result, the intrinsic dimensionality (n/s^2) of the resulting patches should not increase and it is much smaller

[3] A patch of size $\sqrt{n} \times \sqrt{n}$ in $\mathbf{y}_l$ should correspond to a larger patch in $\mathbf{y}_h$, because of the spatial extent of the blur and the scale-up operations. Nevertheless, this additional "band" of pixels can be disregarded, concentrating on predicting only the center portions of the destination patch from $\mathbf{y}_h$.

than the representation dimension (nR), resulting in superfluous computations. The advantage of performing a dimensionality reduction is saving computations in the subsequent training and super-resolution algorithms. Therefore, the last step before turning to the dictionary learning stage is reducing the dimension of the input low-resolution patches, $\{\tilde{\mathbf{p}}_l^k\}_k$. The Principal Component Analysis (PCA) algorithm is applied on these vectors, seeking a subspace on which these patches could be projected while preserving 99.9% of their average energy. The projection operator that transforms the patch $\tilde{\mathbf{p}}_l^k \in \mathbb{R}^{nR}$ to its reduced feature vector, $\mathbf{p}_l^k \in \mathbb{R}^{n_l}$, is denoted by $\mathbf{B} \in \mathbb{R}^{n_l \times nR}$, $\mathbf{p}_l^k = \mathbf{B}\tilde{\mathbf{p}}_l^k$.

3.4 Dictionary Learning

The starting point of the dictionary learning stage are the low-resolution patches $\{\mathbf{p}_l^k\}_k \subseteq \mathbb{R}^{n_l}$. The K-SVD dictionary training procedure [16] is applied to these patches, resulting in the dictionary $\mathbf{A}_l \in \mathbb{R}^{n_l \times m}$:

$$\mathbf{A}_l, \{\mathbf{q}^k\} = \underset{\mathbf{A}_l, \{\mathbf{q}^k\}}{\operatorname{argmin}} \sum_k \|\mathbf{p}_l^k - \mathbf{A}_l \mathbf{q}^k\|^2 \text{ s.t. } \|\mathbf{q}^k\|_0 \leq L \quad \forall k. \tag{8}$$

A side product of this training is the sparse representation coefficients vectors $\{\mathbf{q}^k\}_k$ that correspond to the training patches $\{\mathbf{p}_l^k\}_k$.

The next step is the high-resolution dictionary construction. Recall that our intention is to recover the patch $\mathbf{p}_h^k$ by approximating it as being $\mathbf{p}_h^k \approx \mathbf{A}_h \mathbf{q}^k$. Effectively, the found sparse representation vector for the low-resolution patch is multiplied by the high-resolution dictionary for recovering $\mathbf{p}_l^k$. The dictionary $\mathbf{A}_h$ is therefore sought such that this approximation is as exact as possible. Thus, this dictionary is defined to be the one that minimizes the mean approximation error, i.e.,

$$\begin{aligned} \mathbf{A}_h &= \underset{\mathbf{A}_h}{\operatorname{argmin}} \sum_k \|\mathbf{p}_h^k - \mathbf{A}_h \mathbf{q}^k\|_2^2 \\ &= \underset{\mathbf{A}_h}{\operatorname{argmin}} \|\mathbf{P}_h - \mathbf{A}_h \mathbf{Q}\|_F^2, \end{aligned} \tag{9}$$

where the matrix $\mathbf{P}_h$ is constructed with the high-resolution training patches $\{\mathbf{p}_h^k\}_k$ as its columns, and similarly, $\mathbf{Q}$ contains $\{\mathbf{q}^k\}_k$ as its columns. The solution of the problem is given by the following Pseudo-Inverse expression (given that $\mathbf{Q}$ has full row rank):

$$\mathbf{A}_h = \mathbf{P}_h \mathbf{Q}^+ = \mathbf{P}_h \mathbf{Q}^T (\mathbf{Q}\mathbf{Q}^T)^{-1}. \tag{10}$$

Note that the above approach disregards the fact that the high-resolution patches overlap. Thus, a better (and more complex) training procedure can be envisioned for computing $\mathbf{A}_h$. Since the eventual high-resolution image (in the reconstruction stage) is constructed by positioning high-resolution patches and averaging them, $\mathbf{A}_h$ should be optimized such that the resulting image is as close as possible to the original one.

Define the operator $\mathbf{R}_k$, which extracts a patch of size $n \times n$ from the high resolution image in location k. The image $\hat{\mathbf{y}}_h$ should be constructed by the following formula [10,11]:

$$\hat{\mathbf{y}}_h = \mathbf{y}_l + \left[\sum_{k\in\Omega} \mathbf{R}_k^T \mathbf{R}_k\right]^{-1} \left[\sum_{k\in\Omega} \mathbf{R}_k^T \mathbf{A}_h \mathbf{q}^k\right]. \tag{11}$$

The term $\mathbf{R}_k^T \mathbf{A}_h \mathbf{q}^k$ builds the high-resolution patch, $\mathbf{A}_h \mathbf{q}^k$, and then positions it in the k-th location in the high-resolution image. The term $\mathbf{W} = \sum_k \mathbf{R}_k^T \mathbf{R}_k \in \mathbb{R}^{N_h \times N_h}$ is a diagonal matrix that weights every pixel in the high-resolution outcome, based on the number of contributions it gets form the overlapped patches. Note that $\mathbf{y}_l$ appears in the error computation, due to the fact that the patches in $\mathbf{P}_h$ are constructed from the difference images $\mathbf{e}_h = \mathbf{y}_h - \mathbf{y}_l$, and this means that for the image $\hat{\mathbf{y}}_h$ to be constructed, the algorithm should return these low-frequencies.

Based on the above, it is natural to define the best dictionary $\mathbf{A}_h$ as the solution of the optimization task:

$$\begin{aligned} \mathbf{A}_h &= \arg\min_{\mathbf{A}_h} \|\mathbf{y}_h - \hat{\mathbf{y}}_h\|_2^2 \qquad (12) \\ &= \arg\min_{\mathbf{A}_h} \left\| \mathbf{y}_h - \mathbf{y}_l - \left[\sum_{k\in\Omega} \mathbf{R}_k^T \mathbf{R}_k\right]^{-1} \left[\sum_{k\in\Omega} \mathbf{R}_k^T \mathbf{A}_h \mathbf{q}^k\right] \right\|_2^2. \end{aligned}$$

Denote $\mathbf{W}_k = \mathbf{R}_k \mathbf{W}^{-1} \in \mathbb{R}^{n \times N_h}$ and write $\hat{\mathbf{y}}_h = \mathbf{y}_l + \sum_k \mathbf{W}_k^T \mathbf{A}_h \mathbf{q}^k$. The goal is to minimize $\|\mathbf{y}_h - \hat{\mathbf{y}}_h\|_2^2$ with respect to $\mathbf{A}_h$. Denote $\mathbf{e}_h = \mathbf{y}_h - \mathbf{y}_l$, and $\mathbf{A}_h$ is obtained by the minimization of

$$\mathbf{A}_h = \arg\min_{\mathbf{A}_h} \|\mathbf{y}_h - \mathbf{y}_l - \sum_k \mathbf{W}_k^T \mathbf{A}_h \mathbf{q}^k\|_2^2 = \arg\min_{\mathbf{A}_h} \|\mathbf{e}_h - \sum_k \mathbf{W}_k^T \mathbf{A}_h \mathbf{q}^k\|_2^2. \tag{13}$$

Given $\mathbf{X} \in \mathbb{R}^{n \times m}$, define $\mathbf{x} \equiv \mathbf{cs}(\mathbf{X})$ to be column-stack version of $\mathbf{X}$ (using $\mathbf{x}_{i+nj} = \mathbf{X}_{ij}$). Using the Kronecker product property: $\mathbf{cs}(\mathbf{BAC}) = \left(\mathbf{C}^T \otimes \mathbf{B}\right) \mathbf{cs}(\mathbf{A}) = \left(\mathbf{C} \otimes \mathbf{B}^T\right)^T \mathbf{cs}(\mathbf{A})$, we obtain

$$\mathbf{cs}(\mathbf{e}_h) = \mathbf{e}_h = \sum_k \mathbf{W}_k^T \mathbf{A}_h \mathbf{q}^k = \left(\sum_k \mathbf{q}^k \otimes \mathbf{W}_k\right)^T \mathbf{cs}(\mathbf{A}_h) = \mathbf{M} \cdot \mathbf{cs}(\mathbf{A}_h), \tag{14}$$

where $\mathbf{M} \in \mathbb{R}^{N_h \times mn}$ is defined as $\mathbf{M}^T = \sum_k \mathbf{q}^k \otimes \mathbf{W}_k$. Therefore, one way to get the optimal $\mathbf{A}_h$ is by the direct formula, $\mathbf{M}^\dagger \mathbf{e}_h$.

Since the matrices involved may be too large, we present an alternative, iterative, approach. Note that the gradient of $f(\mathbf{A}_h) = \frac{1}{2}\|\mathbf{e}_h - \sum_k \mathbf{W}_k^T \mathbf{A}_h \mathbf{q}^k\|_2^2$ with respect to $\mathbf{A}_h$, can be written as

$$\nabla_{\mathbf{A}_h} f = \sum_k \mathbf{W}_k \left(\mathbf{e}_h - \sum_k \mathbf{W}_k^T \mathbf{A}_h \mathbf{q}^k\right) \left(\mathbf{q}^k\right)^T. \tag{15}$$

An iterative scheme (such as the Conjugate Gradient method) can be used to find the optimal $\mathbf{A}_h$, using the gradient expression above.

The dictionary resulting from the training process is expected to better reconstruct the output result. In the experiments given below, both ways to derive $\mathbf{A}_h$ are adopted.

The two corresponding dictionaries $\{\mathbf{A}_l, \mathbf{A}_h\}$ conclude the training phase of the super-resolution algorithm, that started with the high-resolution training set $\{\mathbf{y}_h^j\}_j$.

3.5 Reconstruction Phase

The reconstruction phase attempts to magnify a low-resolution image $\mathbf{z}_l$. This image is assumed to have been generated from a high-resolution image $\mathbf{y}_h$ by the same blur and scale-down operations as used in the training. The following steps are performed:

1. The image is scaled up by a factor of s using bicubic interpolation $\mathbf{Q}$, resulting with $\mathbf{y}_l \in \mathbb{R}^{n_l}$.
2. The image $\mathbf{y}_l$ is filtered using the same R high-pass filters that were used for feature extraction in the training, obtaining $f_k * \mathbf{y}_l$.
3. Patches are extracted from these R images, each of size $\sqrt{n} \times \sqrt{n}$ from locations $k \in \Omega$. Every R such patches that correspond to the same location are concatenated to form a patch vector $\tilde{\mathbf{p}}_l^k$. This collection of patches forms the set $\{\tilde{\mathbf{p}}_l^k\}_k$.
4. The patches $\{\tilde{\mathbf{p}}_l^k\}_k$ are multiplied by the projection operator $\mathbf{B}$ for dimensionality reduction, resulting with the set $\{\mathbf{p}_l^k\}_k$, each patch of length n_l (for $n = 81$ and a scale factor $s = 3$, our tests lead to $n_l \approx 30$).
5. The OMP algorithm is applied to $\{\mathbf{p}_l^k\}_k$, allocating L atoms to their representation, and finding the sparse representation vectors $\{\mathbf{q}^k\}_k$.
6. The representation vectors $\{\mathbf{q}^k\}_k$ are multiplied by the high-resolution dictionary $\mathbf{A}_h$, and the approximated high-resolution patches, $\{\mathbf{A}_h\mathbf{q}^k\}_k = \{\hat{\mathbf{p}}_h^k\}_k$ are obtained.
7. The final super-resolved image $\hat{\mathbf{y}}_h$ is constructed from $\hat{\mathbf{p}}_h^k$ by solving the following minimization problem with respect to $\hat{\mathbf{y}}_h$:

$$\hat{\mathbf{y}}_h = \underset{\hat{\mathbf{y}}_h}{\operatorname{argmin}} \sum_k \left\| \mathbf{R}_k(\hat{\mathbf{y}}_h - \mathbf{y}_l) - \hat{\mathbf{p}}_h^k \right\|_2^2 . \tag{16}$$

This problem states that extracted patches from the resulting difference image, $\hat{\mathbf{y}}_h - \mathbf{y}_l$, should be as close as possible to the approximated patches, $\hat{\mathbf{p}}_h^k$. This problem has a closed-form Least-Squares solution, given by

$$\hat{\mathbf{y}}_h = \mathbf{y}_l + \left[\sum_k \mathbf{R}_k^T \mathbf{R}_k \right]^{-1} \sum_k \mathbf{R}_k^T \hat{\mathbf{p}}_h^k, \tag{17}$$

which was already mentioned above. This seemingly complex term is actually very simple – it is equivalent to putting $\hat{\mathbf{p}}_h^k$ in their proper locations, averaging in overlap regions, and adding $\mathbf{y}_l$ to get the final image $\hat{\mathbf{y}}_h$.

3.6 Bootstrapping approach

If the training process has no access to an external set of images, the algorithm may be adapted to train and "bootstrap" itself from a single test image, as proposed by [19]. Note that in order to train the dictionaries $\{\mathbf{A}_l, \mathbf{A}_h\}$, the proposed algorithm needs only access to pairs of low-resolution and high-resolution images. Using the test image $\mathbf{z}_l$ as the "high-resolution" image and its scaled-down version $\mathbf{z}_{ll}$ (by an appropriate choice of $\mathbf{S}$ and $\mathbf{H}$), the algorithm can be easily extended to perform "bootstrapping" from a single image:

- The training phase is applied to $\mathbf{z}_l$ (as high-resolution image), $\mathbf{z}_{ll}$ (as low-resolution image).
- The trained dictionaries are used to enable reconstruction phase, which scales up $\mathbf{z}_l$ into $\mathbf{y}_h$.

Thus, it is possible to scale-up a single image by learning the *Sparse-Land* model directly from the test image itself, provided that the training process has enough training data to build a valid *Sparse-Land* model from.

4 Results

In this section we provide series of tests of the proposed algorithm, and comparisons to the results obtained in [13,14]. In all experiments, the dictionary $\mathbf{A}_h$ is trained using the simpler method (that does not take the overlaps into account), unless mentioned differently.

4.1 Text Scale-Up

The first test contains images showing a printed text. The training image (screen-grabbed) is shown in Figure 3. Only this image is used for the training, and we expect that adding more images would lead to improved results. The global operator $\mathbf{L}_{all}$ in this experiment is implemented by first blurring the high-resolution images $\mathbf{y}_h^j$ with a 1D filter $[1,3,4,3,1]/12$ both horizontally and vertically, and then down-sampling it by a factor of $s = 3$, i.e., the scaled-down image $\mathbf{z}_l$ is one-ninth of the original image size. The image $\mathbf{y}_l$ is created by bicubic interpolation of $\mathbf{z}_l$, returning to the original size.

Extraction of features from the low-resolution images is done exactly as proposed in [13,14] using 4 filters that perform 1-st and 2-nd horizontal and vertical derivatives: $f_1 = [1,-1] = f_2^T$ and $f_3 = [1,-2,1] = f_4^T$. These filters are applied such that only sampled pixels are used in the filtering computation[4]. The patch size used is $n = 81$, and the PCA results with a reduction from $4 \cdot 81 = 324$ dimensions to $n_l = 30$ dimensions. The dictionary training procedure applied

[4] This means that either $\mathbf{z}_l$ is filtered and then interpolated, or $\mathbf{y}_l$ is filtered with zero-padded filters of the form $f_1 = [0,0,1,0,0,-1] = f_2^T$ and $f_3 = [1,0,0,-2,0,0,1] = f_4^T$.

40 iterations of the K-SVD algorithm, with $m = 1,000$ atoms in the dictionary, and allocating $L = 3$ atoms for each representation vector.

The test image (a different image, grabbed from a different page, but having the same scale) is shown in Figure 4. This figure shows the original test image, and the scaled-down version that should be scaled-up. The scaling-up results are also shown in Figure 4, and it is evident that the outcome is far better, compared to the bicubic interpolation, showing Peak-SNR (PSNR) improvement of 2.27dB, with PSNR computed by

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{255^2 \cdot N}{\sum_i (\hat{\mathbf{y}}_i - \mathbf{y}_i)^2} \right), \tag{18}$$

with $\mathbf{y}, \hat{\mathbf{y}} \in [0, 255]^N \subseteq \mathbb{R}^N$.


This book is about *convex optimization*, a special class of mathematical optimization problems, which includes least-squares and linear programming problems. It is well known that least-squares and linear programming problems have a fairly complete theory, arise in a variety of applications, and can be solved numerically very efficiently. The basic point of this book is that the same can be said for the larger class of convex optimization problems.

While the mathematics of convex optimization has been studied for about a century, several related recent developments have stimulated new interest in the topic. The first is the recognition that interior-point methods, developed in the 1980s to solve linear programming problems, can be used to solve convex optimization problems as well. These new methods allow us to solve certain new classes of convex optimization problems, such as semidefinite programs and second-order cone programs, almost as easily as linear programs.

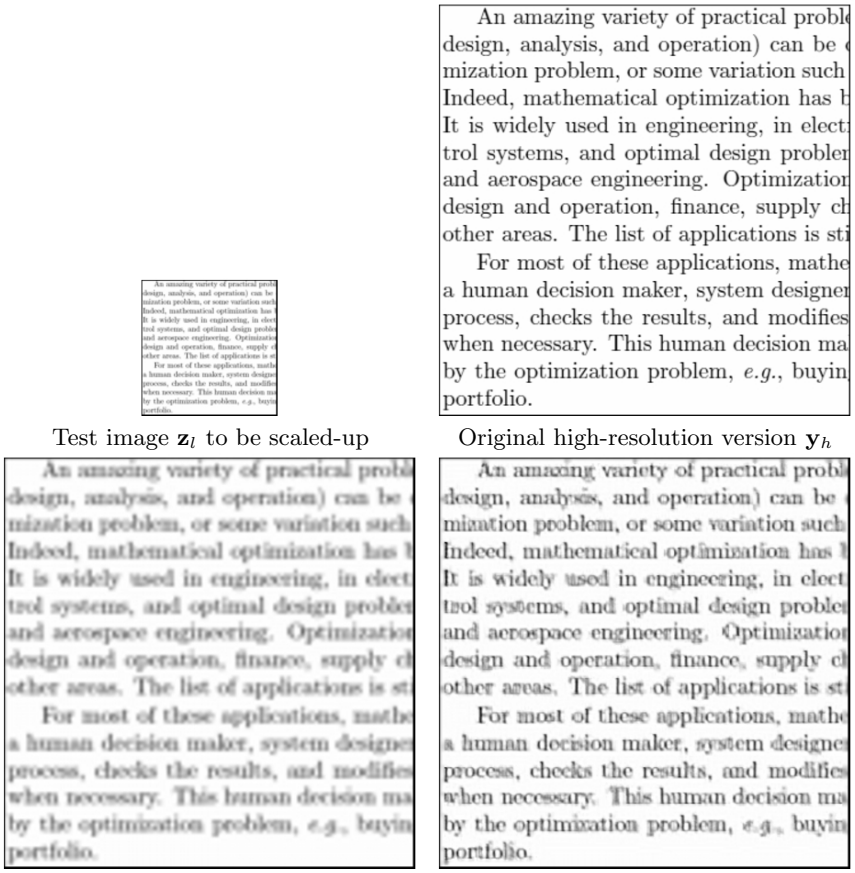
The second development is the discovery that convex optimization problems (beyond least-squares and linear programs) are more prevalent in practice than was previously thought. Since 1990 many applications have been discovered in areas such as automatic control systems, estimation and signal processing, communications and networks, electronic circuit design, data analysis and modeling, statistics, and finance. Convex optimization has also found wide application in combinatorial optimization and global optimization, where it is used to find bounds on the optimal value, as well as approximate solutions. We believe that many other applications of convex optimization are still waiting to be discovered.

There are great advantages to recognizing or formulating a problem as a convex optimization problem. The most basic advantage is that the problem can then be solved, very reliably and efficiently, using interior-point methods or other special methods for convex optimization. These solution methods are reliable enough to be embedded in a computer-aided design or analysis tool, or even a real-time reactive or automatic control system. There are also theoretical or conceptual advantages of formulating a problem as a convex optimization problem. The associated dual

Fig. 3. Text experiment: The training image for the image scaling-up algorithm. This image is of size 717×717 pixels, and it provides a set of $54,289$ training patch-pairs.

4.2 PSNR Comparison with Yang et. al. [13,14]

The second experiment aims to give a comprehensive comparison between the results of our algorithm and the one in [13,14]. The proposed algorithm is implemented in MATLAB using optimized implementation for K-SVD and OMP



Test image $\mathbf{z}_l$ to be scaled-up Original high-resolution version $\mathbf{y}_h$

Bicubic interpolation $\mathbf{y}_l (RMSE{=}47.06)$ Proposed algorithm $\hat{\mathbf{y}}_h (RMSE{=}36.22)$

Fig. 4. Text experiment: The image $\mathbf{z}_l$ is of size 120×120 pixels, and it provides a set of $12,996$ patches to operate on. RMSE is computed using $\frac{1}{N}\sqrt{\sum_{i=1}^{N} |\mathbf{y}_i - \hat{\mathbf{y}}_i|^2}$ where $\mathbf{y}, \hat{\mathbf{y}} \in [0, 255]^N$.

algorithms [17], on Intel Core 2 Duo P8600 at 2.4GHz with 4GB of RAM. It is trained on the same training set used in [13,14], using $s = 3$ scale-up configuration. Each training image is blurred using a bicubic filter and decimated by a factor of s; feature extraction is done as before (using gradient and laplacian filters).

Around 130,000 training patch-pairs are collected and PCA is applied to reduce feature dimensions to $n_l = 30$. Low-resolution dictionary learning takes approximately 12 minutes for 40 iterations of the K-SVD algorithm, with $m = 1,000$ atoms in the dictionary, and allocating $L = 3$ atoms per patch-representation. Moreover, the high-resolution dictionary training takes just a few seconds, using the pseudo-inverse expression $\mathbf{A}_h = \mathbf{P}_h\mathbf{Q}^+$. The proposed training algorithm is much faster than the one used by Yang et. al. [13,14] (taking several hours to run on the same settings). The reconstruction algorithm is tested on 14 test images (taking a few seconds on each image, using fully overlapping 3×3 patches in low-resolution scale) and its results are compared versus bicubic interpolation and the reconstruction algorithm proposed by Yang et. al. [13,14]. The resulting images' boundary is cropped (to ignore boundary effects of overlap-and-add method) and Peak-SNR is computed.

The results are summarized in Table 1. A few 100×100 representative windows from different images are compared at Figure 5. On the left is the original image, followed by bicubic interpolation, Yang et. al. [13,14] and the proposed algorithm's results on the right, at the last column.

Name	Bicubic	Yang et. al [13,14]	Proposed (i)	Proposed (ii)
`baboon`	23.2	23.5	23.5	23.5
`barbara`	26.2	26.4	26.8	26.7
`bridge`	24.4	24.8	25.0	25.0
`coastguard`	26.6	27.0	27.1	27.2
`comic`	23.1	23.9	24.0	24.1
`face`	32.8	33.1	33.5	33.6
`flowers`	27.2	28.2	28.4	28.6
`foreman`	31.2	32.0	33.2	33.6
`lenna`	31.7	32.6	33.0	33.1
`man`	27.0	27.8	27.9	28.0
`monarch`	29.4	30.7	31.1	31.4
`pepper`	32.4	33.3	34.1	34.2
`ppt3`	23.7	25.0	25.2	25.6
`zebra`	26.6	28.0	28.5	28.6
Average	27.5	28.3	28.7	28.8

Table 1. PSNR comparison results. (i) corresponds to the simpler high-resolution dictionary training method, using Pseudo-Inverse expression (see Equation 10). (ii) corresponds to the more complex method that takes care for overlapping patches.

Fig. 5. Visual comparison: Portions from various images (from top to bottom: `barbara`, `comic`, `face`, `pepper`, `zebra`). Left to right: the original image, bicubic interpolation, Yang et. al [13,14] and the proposed algorithm (using the more complex method for dictionary update). Note that the proposed algorithm produces sharper results, preserves the small details of the image and has less visual artifacts compared with bicubic interpolation and Yang et. al. [13,14].

The proposed algorithm performs visually much better than bicubic interpolation, and on some images considerably better than Yang et. al. [13,14] algorithm, having less visual artifacts and producing sharper results with improved PSNR. Moreover, the implementation of the proposed algorithm is much faster (by an order of magnitude) than Yang et. al. [13,14] implementation, using optimized K-SVD and OMP implementations by [17].

4.3 Bootstrapping approach for Single Image Scale-Up

The third experiment is the image `Building`. Starting from the original image $\mathbf{y}_h$ of size 800×800 pixels, the image is filtered with the separable filter $[1, 2, 1]/4$ (horizontally and vertically), and down-scaled by a factor of $s = 2$ to obtain $\mathbf{z}_l$ of size 400×400.

In this experiment, the dictionaries are trained using the very same image, by further scaling it down by a factor $s = 2$, resulting with the image $\mathbf{z}_{ll}$ of size 200×200. The image pair $\{\mathbf{z}_l,\ \mathbf{z}_{ll}\}$ is used for the training, based on the expectation that the relation between these two images reflects also the relation that should be used to up-scale from $\mathbf{z}_l$ to $\mathbf{y}_h$.

Extraction of features from the low-resolution images is done using the same 4 filters, and the dimensionality reduction leads this time from $n = 81$ to $n_l = 42$. The training data contains $37,636$ pairs of low- and high-resolution patches to be modeled. The parameters of the dictionary training all remain the same (40 iterations of the K-SVD algorithm, $m = 1000$ atoms in the dictionary, and $L = 3$ atoms per representation).

Figure 6 shows the original image $\mathbf{y}_h$, the bicubic scaled-up image $\mathbf{y}_l$, and the result of the scaling up algorithm, $\hat{\mathbf{y}}_h$. The difference between the two images is 3.32dB, and in order to see where these differences reside, the figure also shows the difference image $|\hat{\mathbf{y}}_h - \mathbf{y}_h|$. Figure 7 shows two 100×100 portions extracted from $\mathbf{y}_h$, $\mathbf{y}_l$, and $\hat{\mathbf{y}}_h$, to better demonstrate the visual gain achieved by the scaling-up algorithm. This approach has been tested on several images using various dictionary sizes, and in many cases the bootstrapped results are visually comparable and even better, compared to a separate off-line training.

The proposed algorithm results are visually comparable to [19], as demonstrated in Figure 8. Since [19] provides no "ground-truth" images, it is not possible to provide PSNR results. It should be noted that the algorithm of [19] is also much more computationally demanding than the proposed one (requiring the solution of many nearest-neighbor problems for the reconstruction phase) and relies heavily on patch recurrence property. Nevertheless, it does performs quite well on the testing set of images described in [19], assumably due to the "coarse-to-fine" approach (where the image is scaled-up n times, each time by $\sqrt[n]{s}$ factor), especially for large scaling factor s.

When compared to the algorithms [13, 14] and [18], the proposed algorithm uses the same idea of training phase and reconstruction phase, Sparse-land modeling of the desired image patches, and a pair of dictionaries that are used to migrate from the low-resolution domain to the high-resolution one. However, different algorithms are used for the dictionary training: K-SVD for the

The original `Building` image $\mathbf{y}_h$

The difference image $|\hat{\mathbf{y}}_h - \mathbf{y}_h|$ magnified by 5

Bicubic interpolated image $\mathbf{y}_l$ (RMSE= 12.78)

Proposed algorithm's result $\hat{\mathbf{y}}_h$ (RMSE= 8.72)

Fig. 6. Bootstrapping experiment.

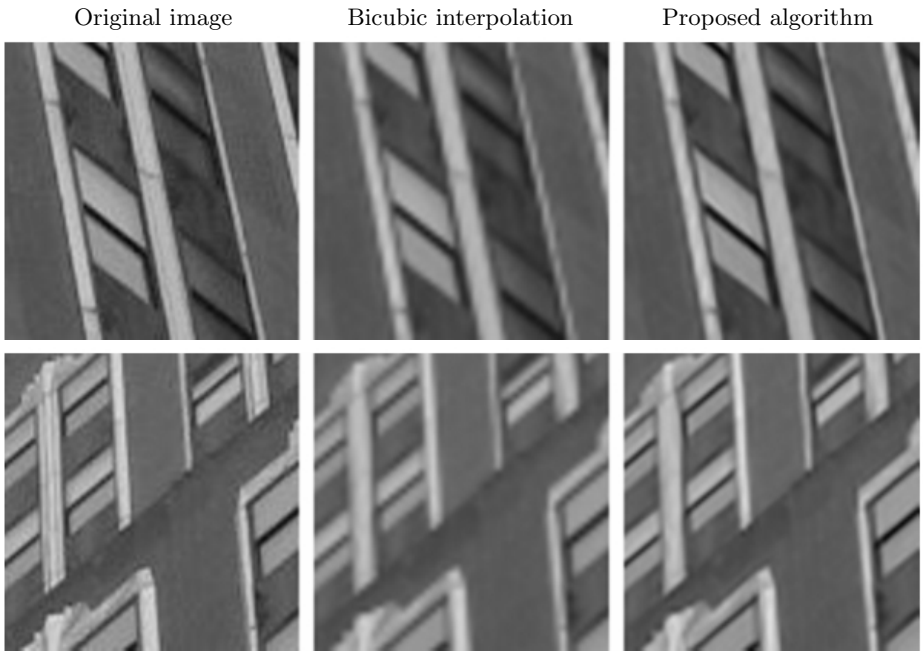


Fig. 7. Bootstrapping experiment: Portions from the `Building` image. Notice that the original portions show some compression artifacts, which do not appear in the scaled-up results.

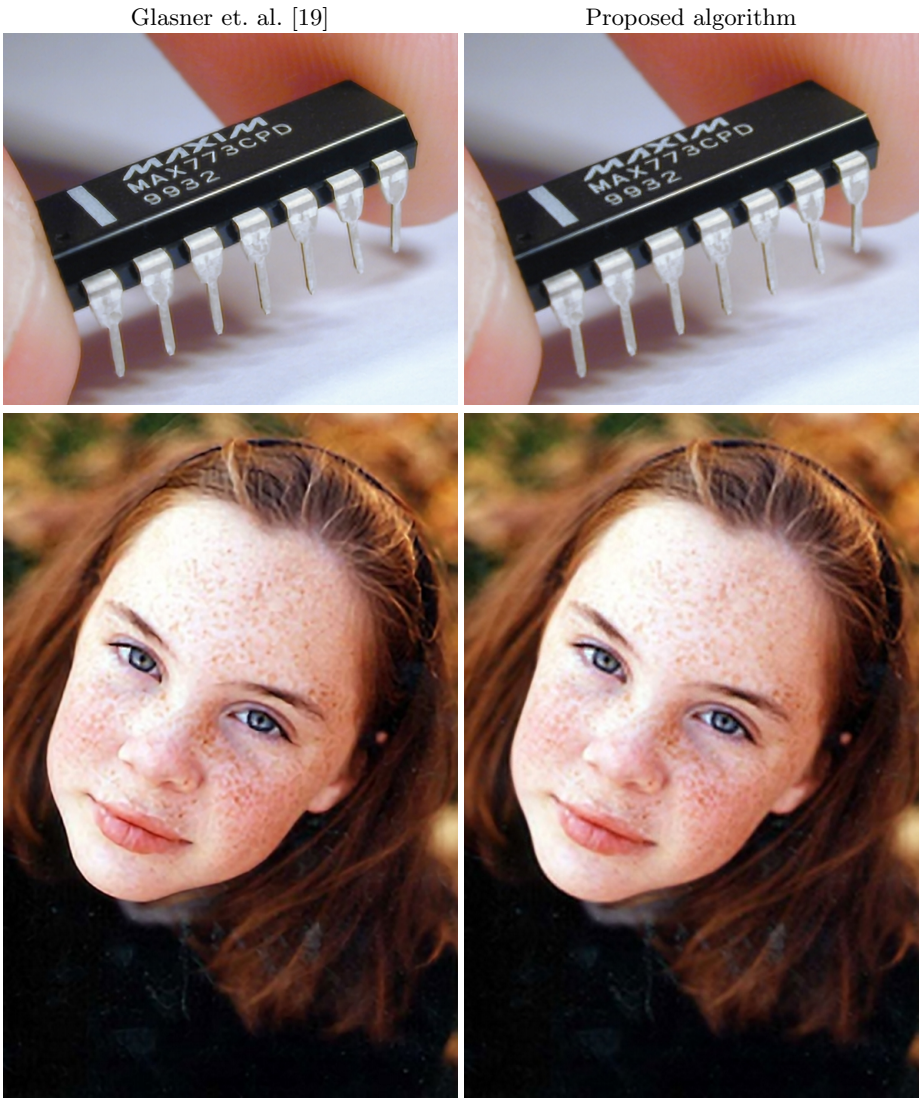


Fig. 8. Visual comparison of the results. First row: Scale-Up by $\times 4$; Second row: Scale-Up by $\times 3$.

low-resolution dictionary, and pseudo-inverse for the high-resolution dictionary. Moreover, OMP is used for sparse-coding, instead of LASSO optimization methods. Other important modifications in our algorithm are (i) the ability to train on the given image, (ii) the initial interpolation by $\mathbf{Q}$ that simplifies much of the subsequent work without a computational cost, (iii) the definition of the high-resolution patches based on the difference $\mathbf{y}_h - \mathbf{y}_l$, and (iv) the dimensionality reduction we apply on the low-resolution patches.

It should be noted that the sparsity constraint $L = 3$ is used, which is much smaller than the sparsity achieved by [13, 14], while the proposed algorithm's PSNR results are better. This may be explained by the better generalization ability of the proposed model, requiring much less atoms per patch. Moreover, while [13, 14] suggests training high-resolution and low-resolution dictionaries together (by concatenating high-resolution and low-resolution patches together into one vector), the proposed process is split as described above - thus achieving a more stable reconstruction process having less visual artifacts.

In [13,14] and in [19] it was observed that the result $\hat{\mathbf{y}}_h$ does not necessarily conform with the requirement $\mathbf{L}_{all}\hat{\mathbf{y}}_h = \mathbf{y}_l$. A back-projection procedure was suggested in which the result $\hat{\mathbf{y}}_h$ is projected onto this constraint, by solving the following optimization problem:

$$\hat{\mathbf{y}}_h = \underset{\hat{\mathbf{y}}_h}{\operatorname{argmin}} \|\hat{\mathbf{y}}_h - \mathbf{y}_h\|_2 \text{ s.t. } \mathbf{L}_{all}\hat{\mathbf{y}}_h = \mathbf{y}_l. \tag{19}$$

However, our tests show that such a projection procedure is not needed, and in fact, it may add some artifacts to the image. Thus, our algorithm does not use this post-processing stage.

5 Summary

There are various ways to scale-up an image while preserving edges and small details. In this paper we introduced one such algorithm that illustrates how sparse representation modeling and dictionary learning can be used for this goal. The algorithm operates by training a pair of low- and high-resolution dictionaries, using either training images or exploiting a lower-resolution version of the same image to be handled. The presented algorithm is based on the method proposed by Yang et. al. [13,14], with several important modifications:

- Numerical shortcuts bring the proposed algorithm to be highly efficient and much faster (using interpolation of the low-resolution image and dimensionality reduction via PCA).
- A different training approach is used for the dictionary-pair: K-SVD for learning $\mathbf{A}_l$ from extracted features, and direct-approach (using pseudo-inverse) for $\mathbf{A}_h$ from error patches.
- The OMP algorithm is used as sparse coding algorithm, which is much faster than ℓ_1-optimization-based methods.
- The proposed algorithm is much simplified by removing redundant steps (e.g. back-projection during post-processing stage).

– The algorithm can operate without a training-set, by boot-strapping the scale-up task from the given low-resolution image. This idea is similar in spirit to the concept posed in [19], but the proposed solution is simpler and yet very effective.

This method is relatively simple, and yet produces a substantial improvement over bicubic interpolation.

Various further improvements can be considered, and these are likely to improve the algorithm's output quality:

– It is possible to force the overlapping patches $\hat{\mathbf{p}}_h^k$ to better align with each other. This can be done by operating sequentially on the incoming patches $\mathbf{p}_l^k$, and when applying the sparse coding stage (to produce $\mathbf{q}^k$), a penalty can be added on the distance between the newly constructed patch, $\hat{\mathbf{p}}_h^k$, and the ones already computed. This has been done in [13,14] and quantifying the benefit of this idea should be done.
– Optimization of the feature extraction and dimensionality reduction operators. Various high-pass filters and thresholds have been tested for the PCA stage, however there must be more options to investigate and perhaps even automatically learned.
– The training set can be extended by adding more examples, by applying simple operators on the input images, e.g. rotation by 90°, reflection, etc..
– It should be noted that it is assumed that the blur operator $\mathbf{H}$ is known for all the experiments that have been performed. In the case it is not known (i.e. while bootstrapping from a single image) there is a significant degree of freedom in choosing $\mathbf{H}$, which obviously will affect the results.
– It is possible to combine the off-line training with the bootstrapping approach by training a general dictionary pair $\{\mathbf{A}_l, \mathbf{A}_h\}$ and applying several more iterations on each new test low-resolution image and its down-scaled version $\{\mathbf{z}_l, \mathbf{z}_{ll}\}$. This two stage process will allow the dictionary to "adapt" the reconstruction process to the specific image to be reconstructed.
– Using more than two scales (the "low" and the "high" ones) in a "coarse-to-fine" framework, as practiced in [19] may help improve the scale-up process by building multi-scale sparse-representation for the image.

Acknowledgements

The authors wish to thank the authors of [13,14] for generously sharing their code and data with them.

References

1. H. S. Hou and H. C. Andrews, "Cubic spline for image interpolation and digital filtering", *IEEE Transactions on Signal Processing, vol. 26, pp. 508-517*, 1978.
2. M. Irani and S. Peleg, "Improving Resolution by Image Registration", *CVGIP: Graphical Models and Image Processing, Vol. 53, pp. 231-239*, May 1991

3. R. R. Schultz and R. L. Stevenson, "A Bayesian Approach to Image Expansion for Improved Definition", *IEEE Transactions on Image Processing, vol. 3, no. 3, pp. 233-242*, 1994.
4. W. T. Freeman, E. C. Pasztor and O. T. Carmichael, Learning Low-Level Vision *International Journal of Computer Vision, 40(1), pp. 25-47*, 2000.
5. X. Li and M. Orchard, "New Edge-Directed Interpolation", *IEEE Transactions on Image Processing, vol. 10, pp. 1521-1527*, 2001.
6. H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding", *IEEE Conference on Computer Vision and Pattern Classification (CVPR), vol. 1, pp. 275-282*, 2004.
7. M. Elad and D. Datsenko, "Example-Based Regularization Deployed to Super-Resolution Reconstruction of a Single Image" *The Computer Journal, Vol. 50, no. 4, pages 1-16* April 2007.
8. J. Sun, Z. Xu, and H. Shum, "Image super-resolution using gradient profile prior", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1-8*, 2008.
9. S. Farsiu, D. Robinson, M. Elad and P. Milanfar, Advances and Challenges in Super-Resolution, *International Journal of Imaging Systems and Technology, Vol. 14, No. 2, pp. 47-57*, special issue on high-resolution image reconstruction, August 2004.
10. M. Elad and M. Aharon, Image denoising via learned dictionaries and sparse representation, *International Conference on Computer Vision and pattern Recognition*, New-York, June 17-22, 2006.
11. M. Elad and M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, *IEEE Trans. on Image Processing* 15(12):3736–3745, December 2006.
12. A.M. Bruckstein, D.L. Donoho, and M. Elad, From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images, *SIAM Review, Vol. 51, No. 1, Pages 34-81*, February 2009.
13. J. Yang, J. Wright, T. Huang, and Y. Ma, Image super-resolution as sparse representation of raw image patches, *IEEE Computer Vision and Pattern Recognition* (CVPR), June 2008.
14. J. Yang, J. Wright, T. Huang, and Y. Ma, Image super-resolution via sparse representation, to appear in *IEEE Trans. on Image Processing*.
15. A.M. Bruckstein, D. L. Donoho, and M. Elad, From sparse solutions of systems of equations to sparse modeling of signals and images, *SIAM review*, 51(1):34–81, February 2009.
16. M. Aharon, M. Elad, and A.M. Bruckstein, The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation, *IEEE Trans. on Signal Processing*, 54(11):4311–4322, November 2006.
17. R. Rubinstein, M. Zibulevsky and M. Elad, Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit, *Technical Report - CS Technion*, April 2008.
18. J. Wang, S. Zhu, and Y. Gong, Resolution enhancement based on learning the sparse association of image patches, *Pattern Recognition Letters*, Volume 31, Issue 1, 1 January 2010.
19. D. Glasner, S. Bagon, and M. Irani, Super-Resolution from a Single Image, *International Conference on Computer Vision* (ICCV), October 2009.