Super-resolution using Neighbor Embedding of Back-projection residuals

Marco Bevilacqua, Aline Roumy, Christine Guillemot
SIROCCO Research team
INRIA
Rennes, France
{marco.bevilacqua, aline.roumy, christine.guillemot}@inria.fr

Marie-Line Alberi Morel
Bell Labs France
Alcatel-Lucent
Nozay, France
marie_line.alberi-morel@alcatel-lucent.com

***Abstract*—In this paper we present a novel algorithm for neighbor embedding based super-resolution (SR), using an external dictionary. In neighbor embedding based SR, the dictionary is trained from couples of high-resolution and low-resolution (LR) training images, and consists of pairs of patches: matching patches (*m-patches*), which are used to match the input image patches and contain only low-frequency content, and reconstruction patches (*r-patches*), which are used to generate the output image patches and actually bring the high-frequency details. We propose a novel training scheme, where the m-patches are extracted from *enhanced back-projected interpolations* of the LR images and the r-patches are extracted from the *back-projection residuals*. A procedure to further optimize the dictionary is followed, and finally nonnegative neighbor embedding is considered at the SR algorithm stage. We consider singularly the various elements of the algorithm, and prove that each of them brings a gain on the final result. The complete algorithm is then compared to other state-of-the-art methods, and its competitiveness is shown.**

***Index Terms*—super-resolution; neighbor embedding; example-based; back-projection**

I. Introduction

Single-image super-resolution (SR) refers to a family of techniques that produce a high-resolution (HR) image from a single low-resolution (LR) input. SR is an inherently ill-posed problem, that needs some prior information to be solved. SR techniques can be broadly classified into two main approaches: inverse problem methods (e.g. [1], [2], [3]), where SR is seen as an ill-posed problem and regularization tools are used to solve it, and machine learning (ML) methods, where the prior information we need is implicitly contained in a dictionary.

Example-based SR algorithms belong to the family of ML methods: the learning process is performed locally, by trying to infer the local HR details through the use of small "examples". For general SR purposes, the examples used consist of patches (sub-windows of image) and the dictionary is therefore formed by pairs of patches. These patches are often called in the literature LR and HR patches, but this can be misleading, as their sizes can be the same. We distinguish, instead, between matching patches (*m-patches*), those ones that we use to match the patches extracted from the LR input image, and reconstruction patches (*r-patches*), those ones that actually contain the high-frequency content, useful to super-resolve the input image. In the SR process, then, the LR input image is divided as well into m-patches, and for each input m-patch we look for good correspondences in the dictionary. The output image is built patch by patch, by using the r-patches corresponding to the m-patches selected. For each single patch reconstruction, the number of patch correspondences considered can be one (e.g. [4]), several, or we may also consider sparse combinations of many patches in the dictionary, as done in the sparse representation SR method of [5].

We talk about *neighbor embedding* (NE) based SR [6], [7], [8], when for each input m-patch we select the best candidates from the dictionary, by a K nearest neighbor search, and we look for a linear approximation of it by using those candidates. In the methods cited above, the weights of the linear approximation are computed by solving a sum-to-one constrained least squares (LS) problem, as done in Locally Linear Embedding (LLE) [9]. In [10], instead, the LLE-based solution is replaced by a nonnegative LS problem; the authors show the benefits of having nonnegative weights.

In the wake of these NE-based algorithms, we propose a new example-based SR algorithm with external dictionary, by combining the nonnegative neighbor embedding of [10] and original training schemes to generate the m-patches and r-patches of the dictionary. In particular, we propose a way to produce "enhanced interpolations" of the LR training images: the LR images are first upscaled via bicubic interpolation, and then iteratively refined with a back-projection procedure. The m-patches are extracted from these enhanced interpolations, whereas the r-patches are taken from the HR residuals. Once the dictionary of pairs of patches is formed in this way, we propose to further optimize it by following the *joint k-means clustering* strategy described in [11]. Differently from the sparse dictionary learning method proposed in [5], that learns atoms suitable for sparse representations, this strategy gives as output a dictionary of real patches.

The rest of the paper is organized as follows. Section II reviews the NE-based approach to SR, and the iterative back-projection method. Then, in Section III, we provide the details of our algorithm, by explaining the different "elements" that compose it. In Section IV we analyze, one by one, the effect of these elements, and provide visual and quantitative results of the complete algorithm, when used for super-resolving images.

II. Neighbor embedding SR using an external dictionary

A. General procedure

Single-image super-resolution (SR) [4] refers to the problem of retrieving a HR output image I_H, given a LR input image I_L, where I_L is supposed to have been generated from I_H according to the following model:

$$I_L = (I_H * b) \downarrow_m , \tag{1}$$

i.e. to have been affected by a blurring process, represented by the convolutional kernel b, and subsequently down-sampled according to a magnification factor m.

Example-based single-image SR aims at reversing the image degradation model (1), by using a dictionary of training examples, usually in the form of patches, $\mathcal{D}$. We call $\mathcal{D}$ an "external dictionary" when the patches that compose it are conveniently derived from a set of external training images. In example-based SR, the LR input image is divided in overlapping patches as well, and the reconstruction of the HR image is done patch by patch, with the help of the examples in the dictionary.

Before the SR algorithm itself can start, therefore, a *training phase*, where the dictionary of examples is formed, is needed. The dictionary is dual ($\mathcal{D} = (\mathcal{X}_d, \mathcal{Y}_d)$), i.e. it consists of patches, which, two by two, form *pairs*. We can distinguish then two kinds of patches:

- Matching patches (*m-patches*) - They contain only low-frequency (LF) information, and thus are used to match the patches of the input image. We indicate this set as $\mathcal{X}_d$.
- Reconstruction patches (*r-patches*) - They contain also high-frequency (HF) information, and thus are used to reconstruct the HR output image. We indicate this set as $\mathcal{Y}_d$.

To sample patches that form an external dictionary, as said, we make use of possibly several training images. For each HR training image J_H, we generate a LR counterpart J_L by following the model (1) ($J_L = (J_H * b) \downarrow_m$). Then, m-patches and r-patches are extracted, respectively from J_L and J_H, or processed versions of them. Patches are considered to form a pair, when coming from corresponding locations in a pair of training images.

Once the dictionary is formed, we can start the proper example-based SR algorithm, where, given a LR image I_L as input, the HR output image I_H is reconstructed patch by patch. We refer specifically to neighbor embedding (NE) based SR methods as those example-based techniques, which involve, at the single patch reconstruction stage, the use of K pairs of patches from the dictionary: for each input patch, K candidates in $\mathcal{X}_d$ are found by performing a nearest neighbor (NN) search, and the corresponding reconstruction patches in $\mathcal{Y}_d$ are used to generate the related output patch.

The basic steps of a NE-based SR algorithm can be summarized as follows. The training phase is supposed to have been performed: we have, then, a set of m-patches $\mathcal{X}_d = \{\mathbf{x}_d^j\}_{j=1}^{N_d}$ and the set of corresponding r-patches $\mathcal{Y}_d = \{\mathbf{y}_d^j\}_{j=1}^{N_d}$.

1) Extract from the input image overlapping m-patches: $\mathcal{X}_t = \{\mathbf{x}_t^i\}_{i=1}^{N_t}$.
2) For each input m-patch $\mathbf{x}_t^i \in \mathcal{X}_t$
   a) Find its K-NN in $\mathcal{X}_d$ in terms of Euclidean distance:
   $$\mathcal{N}_i = \underset{\{\mathbf{x}_k\}_{k=1}^K \in \mathcal{X}_d^K}{\arg\min} \sum_{k=1}^{K} \left\| \mathbf{x}_t^i - \mathbf{x}_k \right\|^2 \tag{2}$$
   b) Find a weighted combination that approximates $\mathbf{x}_t^i$ with the selected neighbors, i.e. compute the K weights $\{w_{ij}\}_{j=1}^K$ such that:
   $$\mathbf{x}_t^i \approx \sum_{\mathbf{x}_d^j \in \mathcal{N}_i} w_{ij} \mathbf{x}_d^j . \tag{3}$$
   c) Apply the same weights for the generation of the corresponding output r-patch $\mathbf{y}_t^i$ with the corresponding neighbors in $\mathcal{Y}_d$:
   $$\mathbf{y}_t^i = \sum_{\mathbf{y}_d^j \in \mathcal{R}(\mathcal{N}_i)} w_{ij} \mathbf{y}_d^j . \tag{4}$$
   where $\mathcal{R}(\mathcal{N}_i)$ indicates the set of r-patches in the dictionary corresponding to the selected neighborhood of m-patches $\mathcal{N}_i$.
3) Once all the r-patches are generated, transform them into pixel-based patches, and combine the obtained patches to form the output image.

In the SR procedure described, there are two main key aspects: how to do the training phase, and so the choice of m-patches and r-patches to use in the algorithm; and the way we compute the weights of each patch combination (step 2b), to approximate a single input m-patch and to generate the corresponding r-patch. We discuss about these two issues, respectively, in Section III-A and III-C.

B. Iterative back-projection

An additional tool, used by several SR algorithms (e.g. [12], [5]), once the output super-resolved image $\hat{I}_H$ is generated, is to "back-project" the obtained image in order to assure it 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

$$\hat{I}_H^{t+1} = \hat{I}_H^t + \left(\left(I_L - \hat{I}_L^t \right) \uparrow_m \right) * p , \tag{5}$$

where the LR image at the 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.

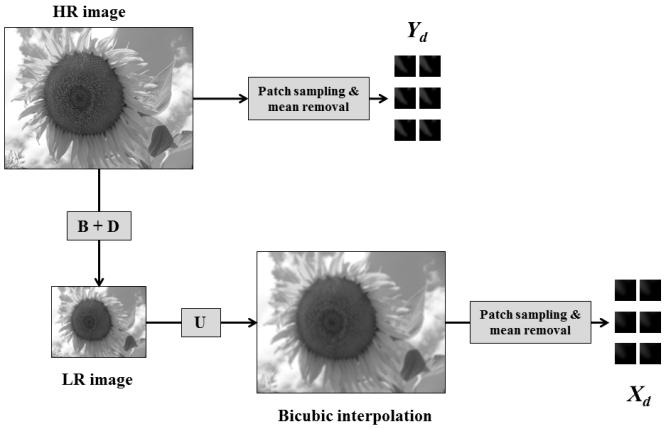


Fig. 1. *Scheme1* for the generation of patches.

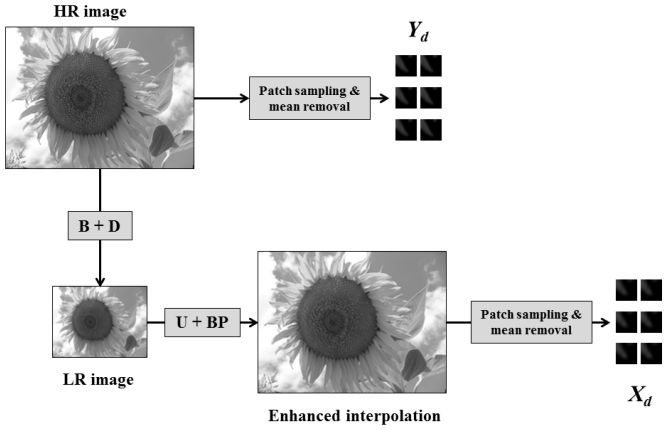


Fig. 2. *Scheme2* for the generation of patches.

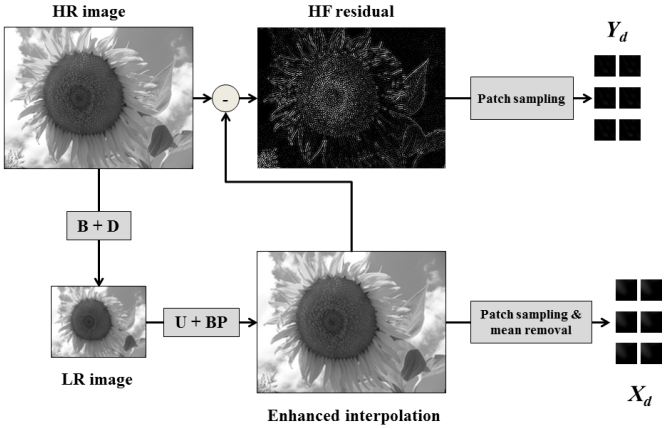


Fig. 3. *Scheme3* for the generation of patches.

III. Proposed algorithm

A. Training phase: patch generation schemes

In example-based SR an important aspect is how the training set is generated. In the case of external dictionaries, as explained in Section II-A, we have a HR external image J_H, we generate its LR counterpart J_L, and from them, or processed version of them, we extract the training patch pairs. Precisely, we extract:

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

As a first scheme (see Fig. 1), as done in [5], we can use directly J_H as a source for r-patches and the bicubic interpolation of J_L, $\mathcal{I}(J_L)$, as a source for m-patches. Therefore, since the two source images have same size, we can sample same-size patches (e.g. 5×5 patches) exactly at the same locations. In [5], the m-patches finally consist of gradient values computed on $\mathcal{I}(J_L)$, whereas the r-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. [10], in fact, shows 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 (Fig. 2) 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 m-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. $\mathcal{E}(J_L)$ does not contain high frequencies as J_H, but it represents a better upscaling of J_L than $\mathcal{I}(J_L)$, and, since m-patches will be similarly produced starting from the input image I_L, a better "starting point" for the SR process.

In [4] 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 [4], we propose instead to use our enhanced interpolation $\mathcal{E}(J_L)$, obtained by bicubic interpolation and back-projection, and compute on that the HF residual image. The new "back-projection residual" $(J_H - \mathcal{E}(J_L))$ is an eaven higher-pass filtered version of J_H, containing really substantial high frequencies. This third learning scheme is represented in Fig. 3.

Table I summarizes the three learning schemes described, specifying for each of them the source for the m-patches and the r-patches. All patches are considered mean-removed, except for the back-projection residuals.

B. Further dictionary learning by joint k-means clustering

The method in [5] is an example-based SR algorithm, where each input m-patch is sparsely coded with the m-patches in the dictionary; the sparse representation found is shared by the r-patches when generating the related output r-patch. As for the

	m-patch source	r-patch source
Scheme1	$\mathcal{I}(J_L)$	J_H
Scheme2	$\mathcal{E}(J_L)$	J_H
Scheme3	$\mathcal{E}(J_L)$	$J_H - \mathcal{E}(J_L)$

TABLE I
m-patches AND *r-patches* IN THE THREE LEARNING SCHEMES PROPOSED.

dictionary, after a large data set is created from natural images, the authors propose a dictionary learning procedure, in order to generate a compact dictionary and impose the m-patches and r-patches to share the same sparse representation. The m-patch and r-patch vectors are concatenated to form a unique matrix, which is the input of the sparse dictionary learning algorithm

A procedure for learning dictionaries to be used with sparse representation methods as in [5] is not suitable for neighbor embedding techniques: since in the algorithm we perform, for the input m-patches, NN searches, we need the patch vectors in the dictionary to stay "real", i.e. composed by real pixel values.

Therefore, we decide to adopt the dictionary learning strategy described in [11]: this method does not arbitrarily create new patches, but identifies in the original dictionary the most salient ones, so reducing its initial size. The main idea is to perform a *joint K-means clustering* (JKC) of the m-patches and the r-patches: a pair of m-/r-patches is placed in a cluster, if both centroids (the m-patch centroid and the r-patch centroid) are the closest ones for the respective patches. As a consequence, it can happen that a pair of patches does not find a placement, because the cluster assignments for the two patches diverge (then, it is temporarily put into a "trash cluster"). In this way, the final clusters are composed by patches, both the m-patches and the r-patches, which are really jointly coherent.

The JKC-based learning method, described in detail in [11], can be summarized in all its steps as follows.

- Take as input a large dictionary of pairs of m-patches and r-patches sampled from natural images
- Perform *JKC* on this data set
- Get rid of the "trash cluster"
- For each cluster, keep only the M closest pairs to the centroid (M prototypes per cluster)
- Apply 8 geometrical transformations to the prototypes

The final geometrical transformations that we apply to the remaining patch pairs are meant to enrich the dictionary with substantial variations on the patch structures. The final size of the dictionary is therefore of $8Mk$ pairs of patches, where k is the chosen number of clusters.

C. Nonnegative neighbor embedding

In the NE procedure, as described in Section II-A, once we create a dictionary of m-patches $\{\mathbf{x}_d^j\}_{j=1}^{N_d}$ and r-patches $\{\mathbf{x}_d^j\}_{j=1}^{N_d}$, we similarly extract m-patches from the input image ($\{\mathbf{x}_t^i\}_{i=1}^{N_t}$) and we super-resolve each of them. For each input m-patch $\mathbf{x}_t^i$ we compute a linear combination of its neighbors in the dictionary in order to approximate it: this is what we call *neighbor embedding*. The weights computed are used to combine the related r-patch neighbors and generate the output r-patch $\mathbf{y}_t^i$.

In the first NE-based SR algorithm [6], which was inspired by Locally Linear Embedding (LLE) [9], the weights were the result of a least squares (LS) approximation problem with a *sum-to-one* constraint. In [10] it is shown that, in a NE scheme with mean-removed patches, in order to avoid over-fitting problems the sum-to-one constraint is better replaced by a *nonnegative* constraint, which represents a milder condition for the weights to be computed. The constrained approximation problem, then, turns into the nonnegative LS (NNLS) problem expressed by the following equation

$$\mathbf{w}^i = \arg\min_{\mathbf{w}} \|\mathbf{x}_t^i - X_d^i \mathbf{w}\|^2 \quad \text{s.t.} \quad \mathbf{w} \geq 0\,, \tag{6}$$

where X_d^i is the matrix of neighbors taken from the dictionary of the input m-patch $\mathbf{x}_t^i$, and $\mathbf{w}^i$ is the related vector of weights.

In our algorithm we use the nonnegative neighbor embedding described above.

D. Overview of the algorithm

Thanks to all the "ingredients" described in the previous sections, we are able to design a complete algorithm for example-based SR. Initially, an external dictionary of m-patches and r-patches is formed by taking training images, alternatively according to one of the three training schemes described in Section III-A. The dictionary so formed is then used as input of the dictionary learning process described in Section III-B: 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 m-patch approximation, we look for the best nonnegative neighbor embedding, as explained in Section III-C.

In Algorithm 1, the nonnegative neighbor embedding SR procedure for the training scheme 3 is reported. We assume that we already have a dictionary of m-patches, as mean-removed patches extracted from enhanced interpolations of LR training images $\mathcal{E}(J_L)$, and a dictionary of r-patches, as the corresponding high-frequency back-projection residual patches.

It is to be noticed that the input image I_L is first upscaled with the enhanced interpolation method described in Section III-A (step 1 of Algorithm 1). The enhanced interpolation becomes therefore our starting point for the SR process, as the m-patches are extracted from it and the actual HR output patches are computed just by summing to them the averaged HF residual ($\mathbf{y}_t^i$). Algorithm 1 is easily adaptable to the other training schemes in Table I, after slight variations.

Algorithm 1 SR by nonnegative NE of back-projection residuals

Input: LR input image I_L, dictionary of *m-patches* $\mathcal{X}_d$, dictionary of *r-patches* $\mathcal{Y}_d$, no. of neighbors K

Output: HR output image $\hat{I}_H$

1: Obtain the enhanced interpolation $\mathcal{E}(I_L)$, by first interpolating and iterative back-projecting w.r.t. I_L
2: Divide $\mathcal{E}(I_L)$ into 5×5 patches (with a 4-pixel overlap)
3: Take the m-patches as mean-removed patches $\mathcal{X}_t = \{\mathbf{x}_t^i\}_{i=1}^{N_t}$
4: Store mean values of the patches $\{\overline{x}_t^i\}_{i=1}^{N_t}$
5: **for** $i = 1 \rightarrow N_t$ **do**
6: Find K-NN in $\mathcal{X}_d$:
$$\mathcal{N}_i = \underset{\{\mathbf{x}_k\}_{k=1}^K \in \mathcal{X}_d^K}{\arg\min} \sum_{k=1}^{K} \left\|\mathbf{x}_t^i - \mathbf{x}_k\right\|^2$$
$$X_d^i = [\mathbf{x}_1, \ldots, \mathbf{x}_K],\ \mathbf{x}_k \in \mathcal{N}_i$$
7: Solve *NNLS*:
$$\mathbf{w}^i = \arg\min_{\mathbf{w}} \|\mathbf{x}_t^i - X_d^i \mathbf{w}\|^2 \quad \text{s.t.} \quad \mathbf{w} \geq 0$$
8: Compute the output r-patches:
$$\mathbf{y}_t^i = \mathbf{w}^i Y_d^i$$
9: **end for**
10: Compute the actual pixel-based HR patches, by adding the computed HF residuals (r-patches) to the input m-patches i.e. $\mathbf{y}_t^i \leftarrow \mathbf{x}_t^i + \overline{x}_t^i + \mathbf{y}_t^i \quad i = 1, \ldots, N_t$
11: Combine together the HR patches to form the HR output image $\hat{I}_H$, by averaging the pixel values in the overlapping regions
12: Iterative back projection to refine $\hat{I}_H$

IV. Experimental results

A. 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 I. In *scheme1* the m-patches are extracted from an interpolated version of the LR image $\mathcal{I}(I_L)$, whereas the r-patches are mean-removed patches taken from the HR image I_H. In *scheme2*, w.r.t. *scheme1*, we just replace the bicubic interpolation with our enhanced interpolation, achieved thanks to an extra back-projection step, $\mathcal{E}(I_L)$. As for the iterative back-projection operation, we choose as back-projection filter p (see equation (5)) a Gaussian kernel of variance $\sigma^2 = 2$. *Scheme3*, in turn, represents a modification of *scheme2*, where we take HF residuals (sampled from $I_H - \mathcal{E}(I_L)$) as r-patches, instead of direct HR patches. 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 *scheme3*, we also add the *JSK*-based dictionary learning procedure described in III-B, 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. 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 III-C.

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

	Bird	Butterfly	Hat	Head	Lena
Scheme1	32.66	25.30	29.27	32.21	29.98
Scheme2	32.98	25.46	29.30	32.30	30.19
Scheme3	32.85	25.37	29.26	32.26	30.13
Scheme3 + JKC	33.37	26.36	29.65	32.28	30.52

TABLE II
RESULTS FOR 5 IMAGES MAGNIFIED BY A **FACTOR OF 3** WITH OUR ALGORITHM AND DIFFERENT TRAINING SCHEMES ADOPTED.

	Bird	Butterfly	Hat	Head	Lena
Scheme1	30.06	22.77	27.71	30.90	28.16
Scheme2	30.41	23.03	27.80	31.05	28.41
Scheme3	30.45	23.12	27.84	31.07	28.45
Scheme3 + JKC	30.41	23.41	27.85	30.92	28.52

TABLE III
RESULTS FOR 5 IMAGES MAGNIFIED BY A **FACTOR OF 4** WITH OUR ALGORITHM AND DIFFERENT TRAINING SCHEMES ADOPTED.

As we can see from the tables, by passing from *scheme1* to *scheme2*, 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 *scheme2* and *scheme3*: the use of HF back-projection residuals instead of mean-removed HR patches seem 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 [11], when applied to *scheme3*. The procedure, as explained in Section III-B, consists initially of a joint k-means clustering of the m-patches and the r-patches; at 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 of [11] (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 [11]. Therefore, more diverging assignments occur, but the pairs of patches that remain at the end are particularly significant. After *JKC*, a patch sampling procedure and a dictionary enrichment with geometric transformations are performed, as described in Section III-B. The final size of the dictionary is about 50000 pairs ($1/10$ of the original one).

In Fig. 4, super-resolved images, related to the methods in Table II, are reported, for the "Butterfly" image magnified

by a factor of 3. The fourth solution (*Scheme3* + 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 of gain.

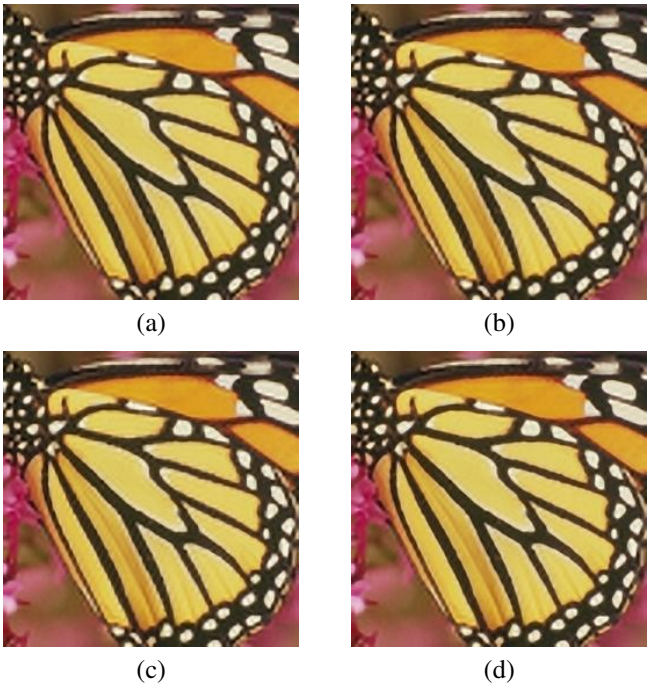


Fig. 4. Visual comparisons on the butterfly image super-resolved by a factor of 3, between the different patch generation schemes: (a) *Scheme1* (b) *Scheme2* (c) *Scheme3* (d) *Scheme3* + *JSK*-based dictionary learning.

B. Visual and quantitative results of super-resolved images

In Section IV-A we showed that *Scheme3* (the enhanced interpolation of the LR image and the back-projection residual used as sources of, respectively, m-patches and r-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 [5], the nonnegative neighbor embedding with centered features of [10], and the pyramid-like algorithm of Glasner et al. [12] (provided by a third-party implementation), which is generally considered among those ones at the top of the state-of-the-art.

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 IV.

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. [5] and the nonnegative NE algorithm with centered features [10], with respect to which it represents a substantial improvement. Moreover, it shows to be highly competitive with the method of Glasner et al. [12], which requires the SR procedure to be repeated several times for smaller scale factors.

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 IV
RESULTS (*PSNR* OF THE SUPER-RESOLVED IMAGE) FOR OUR PROPOSED METHOD AND THREE METHODS IN THE STATE-OF-THE-ART.

The visual results, some of which are reported in Fig. 5, 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).

V. CONCLUSION

In this paper we presented a new algorithm for single-image super-resolution (SR), based on external dictionary and nonnegative embedding. We analyzed three schemes for the patch generation process. In particular, we proposed to use an "enhanced" interpolation for the LR images, obtained by upscaling and refining the related LR image with iterative back-projection (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. We also adopt a joint k-means clustering (*JKC)* procedure to subsequently optimize the dictionary. The *JKC* method, when applied to a dictionary where the low-frequency matching patches and the high-frequency (HF) reconstruction patches are sampled, respectively, from enhanced interpolated LR images and the above mentioned HF residual images, is shown to perform particularly well.

The algorithm so designed has been compared with other single-image SR algorithm at the state-of-the-art. Our proposed algorithm outperforms other one-pass algorithms using external dictionaries, like our previous nonnegative neighbor embedding algorithm [10] and the sparse representation SR method of Yang et al. [5] It also does better than the pyramid-based SR algorithm of Glasner et al [12], considered among the most advanced ones, 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.

As future work, we plan to adapt our designed patch generation schemes, based on enhanced interpolations, and the *JKC*-based dictionary learning procedure, also with other SR techniques.

REFERENCES

[1] Sina Farsiu, Dirk Robinson, Michael Elad, and Peyman Milanfar, "Fast and Robust Multi-Frame Super-Resolution," *IEEE Transactions on Image Processing*, vol. 13, no. 10, pp. 1327–1344, 10 2004.

[2] Shengyang Dai, Mei Han, Wei Xu, Ying Wu, Yihong Gong, and A.K. 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.

[3] Ali Mohammad-Djafari, "Super-Resolution: A Short Review, A New Method Based on Hidden Markov Modeling of HR Image and Future Challenges," *The Computer Journal*, vol. 52, no. 1, pp. 126–141, 2008.

[4] William T. Freeman, Thouis R. Jones, and Egon C. Pasztor, "Example-Based Super-Resolution," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.

[5] Jianchao Yang, J. Wright, T.S. Huang, and Yi Ma, "Image Super-Resolution Via Sparse Representation," *IEEE Transactions on Image Processing,*, vol. 19, no. 11, pp. 2861 –2873, 11 2010.

[6] Hong Chang, Dit-Yan Yeung, and Yimin Xiong, "Super-Resolution Through Neighbor Embedding," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004, vol. 1, pp. 275–282.

[7] Wei Fan and Dit-Yan Yeung, "Image Hallucination Using Neighbor Embedding over Visual Primitive Manifolds," in *2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7 2007, pp. 1–7.

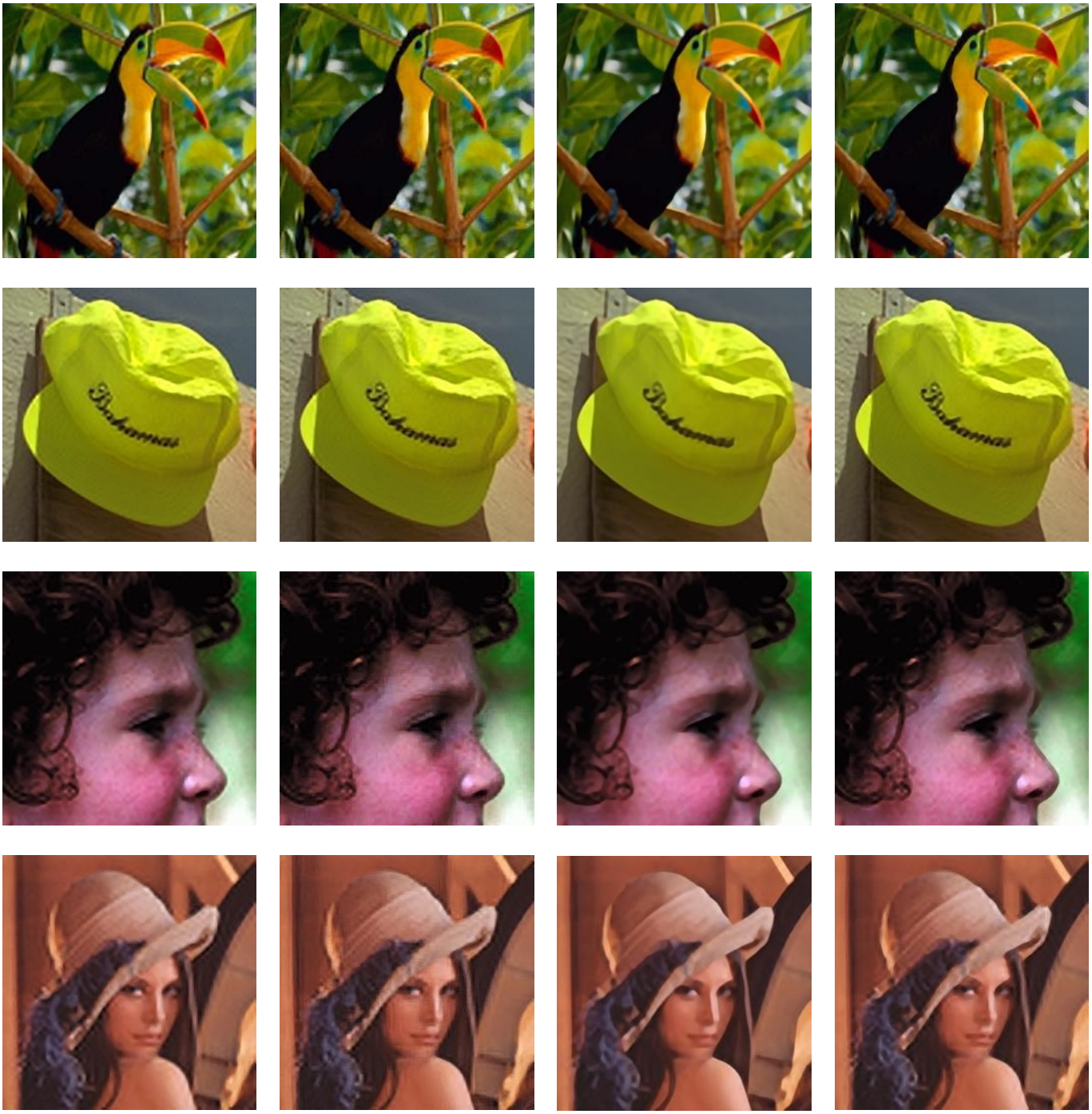
[8] Tak-Ming Chan, Junping Zhang, Jian Pu, and Hua Huang, "Neighbor embedding based super-resolution algorithm through edge detection and feature selection," *Pattern Recognition Letters*, vol. 30, no. 5, pp. 494–502, 4 2009.

[9] Sam T. Roweis and Lawrence K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 12 2000.

[10] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie-Line Alberi Morel, "Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding," in *BMVC (British Machine Vision Conference)*, 2012.

[11] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie-Line Alberi Morel, "Compact and coherent dictionary construction for example-based super-resolution," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

[12] Daniel Glasner, Shai Bagon, and Michal Irani, "Super-Resolution from a Single Image," in *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, 10 2009, pp. 349–356.



(Sparse SR) [5] **(Nonnegative NE)** [10] **(Irani-ICCV2009)** [10] **(Proposed algorithm)**

Fig. 5. Visual comparisons between our proposed algorithm and other three methods in the literature on the following images: **bird x3**, **hat x3**, **head x3**, **lena x4**.