COMPACT AND COHERENT DICTIONARY CONSTRUCTION FOR EXAMPLE-BASED SUPER-RESOLUTION

Marco Bevilacqua[⋆†] *Aline Roumy*[⋆] *Christine Guillemot*[⋆] *Marie-Line Alberi Morel*[†]

[⋆] INRIA Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France
[†] Alcatel-Lucent, Bell Labs France, Route de Villejust, 91620 Nozay, France

ABSTRACT

This paper presents a new method to construct a dictionary for example-based super-resolution (SR) algorithms. Example-based SR relies on a dictionary of correspondences of low-resolution (LR) and high-resolution (HR) patches. Having a fixed, prebuilt, dictionary, allows to speed up the SR process; however, in order to perform well in most cases, we need to have big dictionaries with a large variety of patches. Moreover, LR and HR patches often are not coherent, i.e. local LR neighborhoods are not preserved in the HR space. Our designed dictionary learning 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 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.

Index Terms— Super-resolution, dictionary learning, example-based, neighbor embedding

1. INTRODUCTION

Single-image super-resolution (SR) refers to the task of producing 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. In *example-based* SR [1] the prior information necessary to super-resolve the HR details is implicitly contained in a dictionary of "examples". These examples are intended, in most cases, as correspondences of LR and HR patches, i.e. small squares of pixels. In the SR process, then, also the LR input image is divided into patches. By using the correspondences stored in the dictionary, the output image is built patch by patch, i.e. each patch in the input image is replaced by a HR one, conveniently reconstructed.

Evidently, in example-based SR the choice of the dictionary plays an important role. At this regard, two main kinds of dictionary are possible: external or internal. An external dictionary is formed by sampling HR and LR patches from external training images, respectively some chosen high-definition images and their degraded versions. We speak about an "internal dictionary", instead, when no proper dictionaries are stored, but we learn the LR/HR patch correspondences by putting in relation directly the input image and recursively scaled versions of it, so exploiting the so-called self-similarity property of natural images (e.g [2, 3]).

The clear advantage of having an external dictionary is that it is built in advance: this leads to a reduction on the computational time, whereas in the internal case the dictionary is generated online at each run of the algorithm. Moreover, if we construct the dictionary in a recursive fashion as in [3], its size turns to grow exponentially with the image size: in this case the running time of the algorithm can increase even dramatically, as the neighbor search operation is usually the most time-consuming. On the other hand, a disadvantage of external dictionaries is that they are fixed and so non-adapted to the input image. To be able to satisfactorily process any input, we need to include in the dictionary a large variety of patch correspondences, so turning back to a higher computational time due to the large dictionary size.

In [4], a dictionary learning strategy is proposed, to construct compact dictionaries: under the assumption that a LR patch can be sparsely represented with respect to the patches in the dictionary, new patch vectors are learned as bases for sparse representations. In contrast to [4], we want to design a dictionary learning strategy generally valid for example-based SR algorithms with finite combinations of patches, without relying on the sparse assumption. We propose then a method that addresses as well the problem of creating a compact, yet flexible and efficient, external dictionary for SR purposes, to use as a "passe-partout" for any kind of input image. Our method consists of two steps: a first clustering process that gathers the LR and HR patches of the dictionary into jointly coherent clusters; and a second phase that aims at extracting from the clustered dictionary a set of particularly representative patches that can express the whole dictionary in a compact way. The clustering step is partially similar to the method proposed in [5], where, however, the k-means clustering is done by referring only to the LR patches and is not a joint procedure.

The rest of the paper is organized as follows. Section 2 further discusses the problem of the dictionary in the framework of example-based SR. Then, Section 3 describes our proposed dictionary learning strategy, composed of the two mentioned steps. Before concluding, in Section 4 we validate the dictionary learning strategy, by testing that with an example-based SR algorithm, and present some numerical and visual results.

2. EXAMPLE-BASED SUPER-RESOLUTION WITH AN EXTERNAL DICTIONARY

Single-image example-based SR includes all those learning methods that make use of LR/HR patch correspondences to infer the HR content. We refer specifically to nearest neighbor (NN) based SR methods as those example-based techniques, which involve, at the single patch reconstruction stage, a K-NN search: for each LR input patch K LR candidates in the dictionary are taken into account, and the corresponding HR candidates are used to generate the related output patch. The basic steps of NN-based SR, repeated for each input patch, can be briefly summarized as follows:

1. *NN search*: search for the K best matching LR candidates in the dictionary;

2. *Weight computation*: assign a weight to each of the selected K LR candidates;
3. *HR patch reconstruction*: combine the corresponding HR candidates to reconstruct the HR output patch, according to the weights computed.

Generally, all patches, both in the dictionary and in the target image, are represented as *feature vectors*, i.e. vectors obtained by concatenating certain features computed over their pixel values. In this paper we use our developed algorithm described in [6], based on *nonnegative embedding* and "centered features" (mean-removed luminance values). Here, the weights of each patch combination (step 2) are the result of a nonnegative least squares (LS) problem, which aims at finding the best LS approximation of the related input patch by allowing only additional combinations of patches.

For the sake of designing fast-performing SR algorithms, we decide to use an external dictionary, as also claimed in [6]. Having a fixed dictionary does not require online operations while the SR algorithm is running; however a unique dictionary could not be suitable for any input image. This effect can be observed in Table 1, where the performance of the nonnegative embedding SR algorithm [6] in terms of *PSNR* of the super-resolved image is reported, when tested with different images and 5 equally-sized dictionaries. 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.

		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 1: *PSNR* values of [6], tested with 5 different dictionaries.

In Section 3 we propose a new strategy to build a dictionary, that overcomes the problem of the general non-adaptability of external dictionaries, by building a "general-purpose" dictionary.

3. LEARNING OF A COMPACT AND COHERENT DICTIONARY

An issue with NN-based SR methods, pointed out e.g. in [7], 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 as [8, 6], where the computed weights are meant to capture the local geometry of the patch neighborhoods.

In Section 3.1 we propose a new strategy to limit this problem, based on a *joint k-means clustering* of the initial dictionary. In Section 3.2 we address the problem of designing a compact general-purpose dictionary, thus coming up with a new efficient way of dictionary learning.

3.1. Joint k-means clustering

In order have a neighborhood preservation, and so 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*). [1]

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 define then 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.

To implement this idea we adopt a modified version of the traditional Lloyd's algorithm [9], with the two alternating steps (cluster assignment and re-centering), 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** $\mathbf{c}_{j\prime}^x$ and $\mathbf{c}_{j\prime}^y$ 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: $\mathbf{c}_j^x = \frac{1}{|\mathcal{C}_j|} \sum_{\mathbf{x}_i \in \mathcal{C}_j} \mathbf{x}_i$ and $\mathbf{c}_j^y = \frac{1}{|\mathcal{C}_j|} \sum_{\mathbf{y}_i \in \mathcal{C}_j} \mathbf{y}_i$.
4. Repeat steps 2 and 3 until L no longer changes.

In the procedure described, L is a vector of labels, which keeps track of the vector assignments. We have $L = 0$ 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, as new centers are computed.

3.2. Looking for a compact representation

Table 1, 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 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 sample it, by taking a few *prototypes* per cluster. To do that, we choose to take for each cluster

[1] 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.

the most central elements, i.e. the M closest ones to related center. The sampling is performed in order to select compact and coherent neighborhoods.

As a last step, we consider some simple geometrical transformations of the patches, 3 rotations (for multiples of right angle) and 4 symmetrical reflections (see Fig. 1). These transformations are intended to enrich the dictionary with substantial variations in the patch structures, so balancing the bad effect of the pruning. The prototypes and their transformed versions are finally used to form a new dictionary.

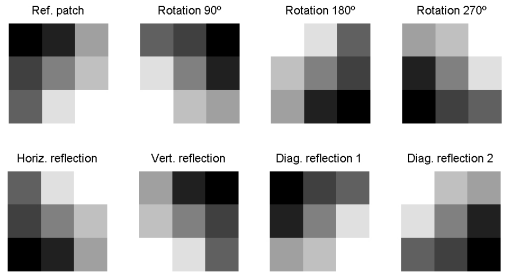

Fig. 1: Geometrical transformations applied to the patches.

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

- Take as input a large dictionary of patches obtained with multi-nature natural image
- Perform *JKC* on this large dictionary
- Keep only M prototypes per cluster
- Apply the geometrical transformations to the prototypes

The size of the dictionary finally created is at most $8Mk$ pairs of patches ("at most" because not all clusters may have at least M elements).

4. EXPERIMENTAL ANALYSIS AND RESULTS

To test the dictionary learning strategy described in Section 3, we start from a large dictionary, concatenation of the 5 dictionaries mentioned in Table 1. 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 *JSK* algorithm with $k = 750$. Some information about the convergence of the *JSK* 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.

As we can see from Table 2, 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 the $M = 12$ most central vectors are kept as "prototypes" of the clusters. 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 nonnegative embedding based SR algorithm of [6], for several images and two different factors (3 and 4). The results for all the dictionaries are reported in Table 3, where "BIG" stands for the concatenated dictionary and "FINAL" for the final dictionary we learn from it with our method (*JSK* + prototype selection).

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: Prospect of convergence and other statistics of the *JSK* algorithm applied to a dictionary of 280K pairs.

		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 3: *PSNR* values of [6], tested with 5 different dictionaries, the concatenation of them, and our designed dictionary.

As we can see from Table 3, the new constructed dictionary performs better than any input dictionary almost overall, 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 Fig. 2, 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 is learned from, 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 4. 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*.

By looking at Table 4, 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

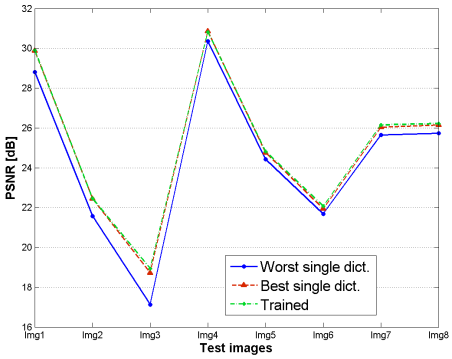

Fig. 2: Performance of our constructed dictionary w.r.t. the best and worst single dictionary for any test image considered.

			JKC			k-means	
Image	**Scale**	**BIG**	**Prun.**	**Sampl.**	**FINAL**	**Sampl.**	**FINAL**
dict. size $\rightarrow$		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 4: Evolution of the *PSNR*, from the big dictionary to the one finally constructed, for *JKC* and standard k-means applied to the concatenated LR-HR vectors.

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.

Finally, Fig. 3 reports some visual results for the "Bird", "Butterfly", and "Head" images, all magnified by a factor of 3. Our nonnegative embedding based algorithm [6] is compared with the pyramid-based algorithm of [3], 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 [3], we report the outputs obtained thanks to a third-party implementation of the algorithm. The nonnegative embedding based algorithm, instead, is tested with one of the input dictionaries of Table 3 and the final dictionary learned with the proposed strategy.

As we can see from the figure, the use of the new dictionary appreciably improves the visual results of the nonnegative embedding SR 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 [3], which presents generally less blurred but somehow unnatural images. On the other hand, the algorithm in [3] requires much higher computational times (see Table 2 in [6]).

5. CONCLUSION

In this paper we described a novel method to construct an external dictionary for single-image super-resolution (SR) purposes. The method is performed totally off-line, i.e. once the dictionary is created it is ready to be used, and involves two steps: the joint k-means clustering (*JKC*), a joint clustering of the LR and HR patches of the dictionary that represents an original variation to the well-known k-means algorithm, and a selecting phase, where only the essential patches are kept (the "prototypes" of each cluster). When used with the NN-based SR algorithm of [6], the so-learned dictionaries are

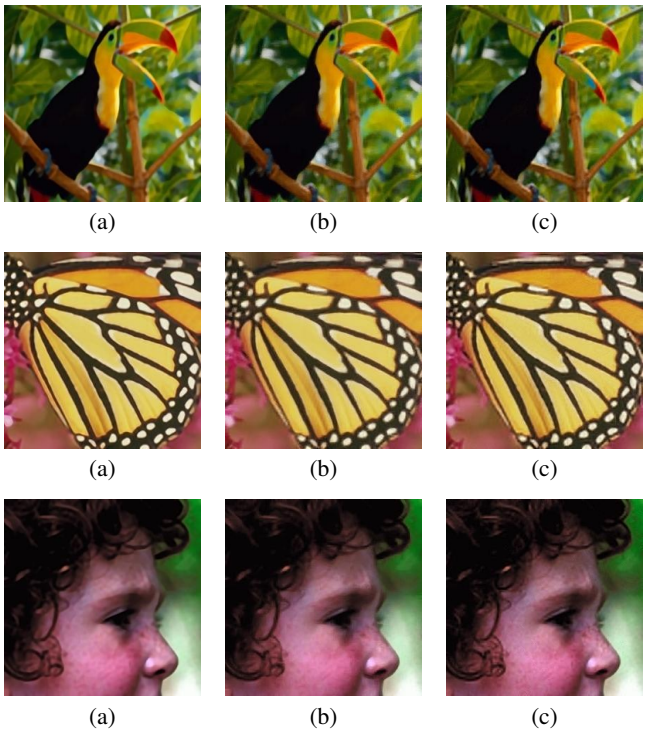

Fig. 3: Visual comparisons between the pyramid-based algorithm of [3] (a) and the nonnegative embedding algorithm [6] with the "Esa" dictionary (b) and the final dictionary (c).

showed to perform generally well, independently of the input image. With comparable dictionary sizes, they perform better than any single non-processed external dictionary. Moreover, they help filling the gap between the one-pass NN-based SR algorithms that make use of external dictionaries and more sophisticated SR algorithms, as [3], which rely on image self-similarities. As for the possible future work, we plan to work on a formal proof for the convergence of the *JKC* algorithm, which we believe possible under certain assumptions.

6. ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and suggestions to consider some other aspects (i.e. the comparison of *JSK* with k-means applied on concatenated vectors). This also led to a better comprehension of the proposed algorithm, and improved the quality of the paper.

7. REFERENCES

[1] 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.

[2] Gilad Freedman and Raanan Fattal, "Image and Video Upscaling from Local Self-Examples," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–10, 2010.

[3] 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.

[4] 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.

[5] Shin-Cheol Jeong and Byung Cheol 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.

[6] 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.

[7] Bo Li, Hong Chang, Shiguang Shan, and Xilin Chen, "Locality preserving constraints for super-resolution with neighbor embedding," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, 11 2009, pp. 1189–1192.

[8] 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.

[9] Stuart P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982.