A+: Adjusted Anchored Neighborhood Regression for Fast Super-Resolution

Radu Timofte, Vincent De Smet, Luc Van Gool

CVL, D-ITET, ETH Zürich, Switzerland
VISICS, ESAT/PSI, KU Leuven, Belgium

Abstract. We address the problem of image upscaling in the form of single image super-resolution based on a dictionary of low- and high-resolution exemplars. Two recently proposed methods, Anchored Neighborhood Regression (ANR) and Simple Functions (SF), provide state-of-the-art quality performance. Moreover, ANR is among the fastest known super-resolution methods. ANR learns sparse dictionaries and regressors anchored to the dictionary atoms. SF relies on clusters and corresponding learned functions. We propose A+, an improved variant of ANR, which combines the best qualities of ANR and SF. A+ builds on the features and anchored regressors from ANR but instead of learning the regressors on the dictionary it uses the full training material, similar to SF. We validate our method on standard images and compare with state-of-the-art methods. We obtain improved quality (*i.e.* 0.2-0.7dB PSNR better than ANR) and excellent time complexity, rendering A+ the most efficient dictionary-based super-resolution method to date.

1 Introduction

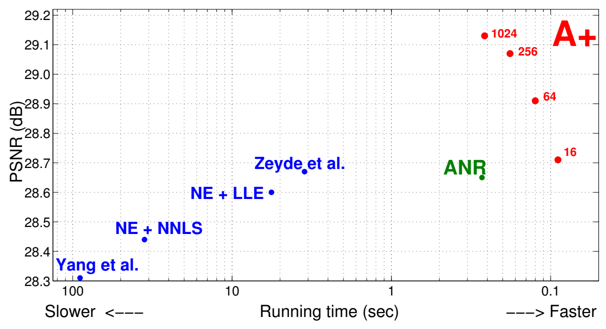


Fig. 1. Our proposed A+ method (*red*, operating points for 16, 64, 256, and 1024 sized dictionaries) provides both the best quality and the highest speed in comparison with state-of-the-art example-based SR methods. A+ preserves the time complexity of ANR (*green*) [1]. See Table 1 and Fig. 2 for more details.

Single-image super-resolution (SR) is a branch of image reconstruction that concerns itself with the problem of generating a plausible and visually pleasing high-resolution (HR) output image from a low-resolution (LR) input image. As opposed to the similar branch of image deblurring, in SR it is generally assumed that the input image, while having a low resolution in the sense of having a low amount of pixels, is still sharp at the original scale. SR methods are mainly concerned with upscaling the image without losing the sharpness of the original low-resolution image.

SR is an ill-posed problem because each LR pixel has to be mapped onto many HR pixels, depending on the desired upsampling factor. Most popular single-image SR methods try to solve this problem by enforcing natural image priors based on either intuitive understanding (*e.g.* natural images consist mainly of flat regions separated by sharp edges) or statistical analysis of many natural images [2–5]. Some recent approaches try to shift the focus from finding good image priors to finding an appropriate blur kernel [6, 7]. In both cases the authors usually work on the level of small image patches. These provide a good basis for finding effective local image priors and can be combined in different ways ranging from simple averaging of overlapping patches to finding maximum-a-posteriori patch candidate combinations using belief propagation or graph cuts [8]. Very recently, convolutional neural networks were applied to this problem [18].

Patch-based SR methods tend to require a large database of many image patches in order to learn effective priors and to super-resolve general classes of natural images. Both the resulting time-complexity and memory requirements can be strong limiting factors on the practical performance of these methods. One class of SR approaches that tries to solve this bottleneck are the neighbor embedding approaches [9, 10]. These have the nice feature of compensating for the lack of density in the patch feature space by assuming that all patches lie on manifolds in their respective LR and HR spaces, which allows for an input patch to be approximated as an interpolation of existing database patches, with one set of interpolation coefficients being applied to both spaces. Another class of methods is focused on creating sparse representations of large patch databases [4, 11], which can reduce overfitting to training data and is a good way of avoiding the need for a large patch database.

Two recent neighbor embedding approaches, ANR [1] and SF [12], have been successful in reducing the time complexity of single-image super-resolution significantly without sacrificing the quality of the super-resolved output image. They show results which are qualitatively and quantitatively on par with other state-of-the-art methods, while improving execution speed by one or two orders of magnitude. We take these approaches (specifically ANR) as a starting point to introduce a novel SR method, which we have dubbed A+, that makes no sacrifices on the computational efficiency and achieves an improved quality of the results which surpasses current state-of-the-art methods. Fig. 1 shows the performance of our method compared to other neighbor embedding and sparsity-based methods and shows our improvement over the original ANR approach.

In the following section we will first give some background on other neighbor embedding approaches and sparse coding approaches and review the SF method. In section 3 we introduce our A+ approach and explain it in detail. Section 4 describes our experiments, where we compare the performance of our approach to other state-of-the-art methods based on quality and processing time. Finally in section 5 we conclude the paper.

2 Dictionary-based Super-Resolution

Our proposed approach builds on theories from neighbor embedding and sparse coding super-resolution methods. Both of these are dictionary-based approaches, which means they rely on a dictionary of patches or patch-based atoms that can be trained to form an efficient representation of natural image patches. This gives them the potential to drastically reduce the computational complexity of patch-based single-image super-resolution methods and enhance their representational power.

2.1 Neighbor embedding approaches

One way to add more generalization ability to the basic patch-based super-resolution scheme is to allow LR input patches to be approximated by a linear combination of their nearest neighbors in the database. Neighbor embedding (NE) approaches assume that the LR and HR patches lie on low-dimensional nonlinear manifolds with locally similar geometry. Assuming this, the same interpolation coefficients that are used between LR patches to approximate an input patch can be used in HR space to estimate an output patch. Chang *et al.* [9] assume that the manifolds lie on or near locally linear patches and use Locally Linear Embedding (LLE) [13] to describe LR patches as linear combinations of their nearest neighbors, assuming the patch space is populated densely enough. The same coefficients can then be used to perform linear interpolation of the corresponding HR patches:

$$\mathbf{x} = \sum_{i=1}^{K} {w^\star}_i {\mathbf{x}'}_i. \tag{1}$$

We use $\mathbf{x}$ to refer to the HR output patch, $\mathbf{x'_i}$ is the i'th candidate HR patch corresponding to the i'th nearest neighbor of the input patch in LR space, ${w^\star}_i$ is the weight of the i'th candidate, and we limit our search to K nearest neighbors. Bevilacqua *et al.* [10] also use neighbor embedding for SR. They use a nonnegative least-squares decomposition to find weights that can be used in LR and HR space.

2.2 Sparse coding approaches

Instead of using a dictionary consisting of a collection of patches taken from natural images, as neighbor embedding methods typically do, one could try to create an efficient representation of the patch space by training a codebook of dictionary atoms. Yang *et al.* [4] start from a large collection of image patches and use a sparsity constraint to jointly train the LR and HR dictionaries so that they are able to represent LR patches and their corresponding HR counterparts using one sparse representation. Once the dictionaries are trained, the algorithm searches for a close sparse representation of each input patch as a combination of dictionary atoms:

$$\min_{\boldsymbol{\alpha}} \|\mathbf{D}_l \boldsymbol{\alpha} - \mathbf{y}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1, \tag{2}$$

where $\mathbf{D}_l$ is the LR dictionary, α is a weight matrix that functions as a sparse selector of dictionary atoms, and λ is a weighing factor to balance the importance of the sparsity constraint. Other approaches, most notably Zeyde *et al.* [11], build on this work and reach significant improvements both in speed and output quality.

2.3 Simple Functions

The neighbor embedding approaches use a database of patches and represent each LR input patch as a combination of its nearest neighbors, whereas the sparse coding approaches explicitly enforce sparsity to create a coupled LR-HR dictionary which is used to map an LR patch to HR space. Another approach would be to cluster the LR patch space and to learn a separate mapping from LR to HR space for each cluster. This is what Yang and Yang [12] propose. They collect a large amount of natural images to harvest patches. These are clustered into a relatively small number of subspaces (*e.g.* 1024 or 4096) for which a simple mapping function from LR to HR is then learned. The authors compare three different functions for this mapping: an affine transformation learned for each cluster using a least squares approximation, and a support vector regressor with either a linear kernel or a radial basis function kernel. Because of the visual similarity of the resulting images for all of these, the authors propose to use the affine transformation, as it is by far computationally the fastest. The mapping coefficients can be learned offline and stored for each cluster. LR patches are then super-resolved by finding the closest cluster center and applying the corresponding transformation,

$$\mathbf{x} = \mathbf{C}_i^\star \begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix}, \ \textit{with}\ \mathbf{C}_i^\star = \arg\min_{\mathbf{C}_i} \left\| \mathbf{Y}_i - \mathbf{C}_i \begin{bmatrix} \mathbf{X}_i \\ \mathbf{1} \end{bmatrix} \right\|_2^2 . \tag{3}$$

We denote with $\mathbf{y}$ and $\mathbf{x}$ an LR input patch that is matched to cluster i and its estimated HR output patch, with $\mathbf{C}_i$ the transformation matrix and $\mathbf{Y}_i$ and $\mathbf{X}_i$ the training patches for cluster i. $\mathbf{1}$ is a vector with the same number of elements as the amount of training patches in $\mathbf{X}_i$, filled entirely with ones. Storing the mapping coefficients results in a big boost in performance speed.

2.4 Anchored Neighborhood Regression

The Anchored Neighborhood Regression approach proposed by Timofte *et al.* [1] has a similar strategy to boost performance speed but shares more properties with sparse coding than the simple functions approach. It relies on precalculating and storing transformations to dramatically improve execution speed at test-time. Starting from the same dictionaries as Zeyde *et al.* [11], which are efficiently trained for sparsity, ANR reformulates the patch representation problem from eq. (2) as a least squares regression regularized by the l_2-norm of the coefficient matrix (which we refer to as $\boldsymbol{\beta}$ to avoid confusion with the sparse methods described in Section 2.2):

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{N}_l \boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2. \tag{4}$$

Instead of considering the whole dictionary like the sparse encoding approach in Section 2.2, the authors propose to work in local neighborhoods $\mathbf{N}_{l,h}$ of the dictionary. The main advantage of working with the l_2-norm is that this turns the problem into Ridge Regression [14] and gives it a closed solution, which means they can precalculate a projection matrix based on the neighborhood that is considered. An LR input patch $\mathbf{y}$ can be projected to HR space as

$$\mathbf{x} = \mathbf{N}_h (\mathbf{N}_l^T \mathbf{N}_l + \lambda \mathbf{I})^{-1} \mathbf{N}_l^T \mathbf{y} = \mathbf{P}_j \mathbf{y}, \tag{5}$$

with $\mathbf{P}_j$ the stored projection matrix for dictionary atom $\mathbf{d}_{lj}$. Each dictionary atom has its own neighborhood $\mathbf{N}_l$ of dictionary atoms assigned to it. The SR process for ANR at test time then becomes mainly a nearest neighbor search followed by a matrix multiplication for each input patch. On the extreme part of the spectrum the neighborhoods could be made the size of the entire dictionary $\mathbf{D}_{l,h}$. In this case each atom has the same transformation matrix and there is no need to perform a nearest neighbor search through the dictionary. This is called Global Regression (GR), and at test time the global transformation can be applied, reducing the process to the application of a simple stored projection. This makes the process lose a lot of flexibility and results in a lower output quality for the benefit of greatly enhanced speed.

3 Adjusted Anchored Neighborhood Regression (A+)

We propose a method based on the ANR approach [1]. We name our method A+ due to the fact that it inherits and adjusts the ANR key aspects such as the anchored regressors, the features, and the test time complexity, while at the same time significantly improving over its performance. We will first explain the insights that lead to A+, then we will provide its general formulation and then discuss the influence of the different model parameters.

3.1 Insights

Most neighbor embedding methods (exceptions are ANR [1] and SF [12]) rely on extracting a neighborhood of LR training patches for each LR test patch, followed by solving the HR patch reconstruction problem, usually through some optimization process. ANR instead places this computation at the training stage, leaving at test time only the search of a neighboring anchor followed by a projection to HR space. The anchoring points are the atoms of a sparse dictionary trained to span the space of LR patch features and to provide a uniform coverage, while at the same time being optimized to have a low decomposition error for the LR patch features available in the training database. On these anchoring points ANR learns offline regressors to the local neighborhood of correlated atoms from the same sparse dictionary.

We make the following observations related to the ANR formulation:

(i) the atoms are rather sparsely sampling the space, while the training pool of samples is (or can be) practically infinite, as long as enough training images are available to harvest patches;
(ii) the local manifold around an atom, which is the associated hypercell of an atom, is more accurately spanned by dense samples interior to the hypercell than by a set of external neighboring atoms.

Therefore, we can expect that the more samples we use from the interior of the anchor hypercell the better the approximation of that subspace or manifold will be, both on the unit hypersphere where the atom lies and on any translation of it along the atom's direction.

For this purpose we will consider the neighborhood of training samples that may cover both the hypercell of the anchoring atom and part of its adjacent atom hypercells. This is one of the key ideas exploited in our A+ method. Another one is the assumption that in the neighborhood of an atom one can regress linearly to a solution that accommodates all the local neighborhood training samples and moreover, can interpolate to samples from the space spanned by them. The bigger the neighborhood of LR patches, the stronger the linearity imposed not only on LR patches but on their corresponding HR patches. The one LR patch to many HR patches problem is tackled by the power of linear LR patch decomposition over many close to unit norm patches from anchored neighborhoods. Having a dictionary of anchors of unit l_2 norm reduces the general clustering of the whole LR space to a clustering around the unit hypersphere, where the atoms are definitive and the correlation of a sample to the atom can determine its adherence to the atom's spanned space of LR patches. Therefore, for a new LR patch, we first find its most correlated atom and then we apply the sample-based regression stored by that atom to reconstruct the HR patch in the same fashion as ANR.

The SF [12] method differs from A+ (and ANR) in at least 3 main aspects: SF uses different LR patch features, follows the Euclidean space clustering assumption and its functions are bounded to the cluster samples. Since a sample can be found on the boundary area between clusters, we consider that either the number of clusters should be consistently large or the functions should be anchored on clusters but learned using also the neighboring clusters. Therefore, while we formulate our insight using the ANR framework, a derivation inside the SF framework could also be possible.

3.2 Formulation

We adopt the dictionary training method of Zeyde *et al.* and ANR, which first optimizes over LR patches to obtain a sparse dictionary $\mathbf{D}_l$ to then reconstruct its corresponding $\mathbf{D}_h$ by enforcing the coefficients in the HR patch decompositions over $\mathbf{D}_h$ to be the same coefficients from the corresponding LR patch decompositions over $\mathbf{D}_l$.

ANR and the other sparse coding approaches have no need for the training samples anymore after the dictionary is trained, but for A+ they are still crucial, as the neighborhood used for regression (calculated during training and used at test time) is explicitly taken from the training pool of samples. We reuse the ridge regression formulation of ANR as shown in eq. (4) to regress at train time, but redefine the neighborhood in terms of the dense training samples rather than the sparse dictionary atoms. Our optimization problem then looks like this,

$$\min_{\boldsymbol{\delta}} \|\mathbf{y} - \mathbf{S}_l \boldsymbol{\delta}\|_2^2 + \lambda \|\boldsymbol{\delta}\|_2. \tag{6}$$

We have replaced the neighborhood of atoms $\mathbf{N}_l$ (of which each dictionary atom has its own version) with a matrix $\mathbf{S}_l$, containing the K training samples that lie closest to the dictionary atom to which the input patch $\mathbf{y}$ is matched. We set K to 2048 in our experiments. The distance measure used for the nearest neighbor search is the Euclidean distance between the anchor atom and the training samples. More details on the sampling can be found in the next section.

3.3 Sampling anchored neighborhoods

In order to have a robust regressor anchored to an atom, we need to have a neighborhood of samples (in a Euclidean sense) centered on the atom, or brought to unit l_2 norm, on the surface of the unit hypersphere. When the l_2 norm of the LR patch feature is below a small threshold (0.1 in our case) we do not l_2-normalize it, as we want to avoid enhancing the potential noise of a flat patch. The local manifold on the unit hypersphere can be approximated by the set of neighboring samples, even if they are not all lying on the hypersphere. However, due to our choice of LR patch features (the same as ANR and the approach of Zeyde *et al.*) we do have a uniform scaling factor between the LR feature and its HR patch. Therefore, when we bring the LR patch features to the hypersphere by l_2 normalization we also transform the corresponding HR patches linearly by scaling them with the same factor (the l_2-norm of the original LR patch features) to preserve the relation between LR and HR spaces.

We propose that if more samples are closer to the anchoring atom the local manifold approximation through regression on these samples will improve. Therefore, we retrieve for each atom as many training samples as a given neighborhood size. The same samples can be shared among different atom centered neighborhoods. This ensures that the regressors are learned robustly even in extreme cases where either the number of atoms is very small with respect to the number of training samples or where we only have few samples around certain atoms.

4 Experiments

In this section we analyze the performance of our proposed A+ method in relation to its design parameters and benchmark it in quantitative and qualitative comparison with ANR and other state-of-the-art methods. [1]

4.1 Benchmarks

We adopt the testing benchmarks of the original ANR algorithm [1] and in addition we use the 100 test images from the BSDS300 Berkeley dataset which is widely used as a benchmark for various computer vision tasks including super-resolution.

Training dataset We use the training set of images as proposed by Yang *et al.* [4] and used, among others, by Timofte *et al.* [1] and by Zeyde *et al.* [11].

Set5 and Set14 'Set5' [10] and 'Set14' [11] contain 5 and respectively 14 commonly used images for super-resolution evaluation. They are used in the same settings as in [1]. In order to compare with ANR as fairly as possible, we conduct most of our experiments related to the internal parameters of our A+ method on Set14. This is similar to ANR, its internal parameters being first evaluated on Set14.

B100 aka Berkeley Segmentation Dataset The Berkeley Segmentation Dataset (BSDS300) [15] is a dataset of natural images which was originally designed for image segmentation but has been widely used in many image restoration approaches such as super-resolution and denoising to test performance [17]. We will use its 100 testing images (named here 'B100') to compare our method more thoroughly to the closely related ANR and GR methods.

Compared Methods We compare with all the methods of [1] under the same conditions as originally compared with ANR. Where applicable, we use a shared sparse dictionary among the methods, or at least the methods use similar sized dictionaries and share the training material, which is the same from [16, 11, 1]. The methods are as follows: NE+LS (Neighbor Embedding with Least Squares), NE+LLE (Neighbor Embedding with Locally Linear Embedding, similar to Chang *et al.* [9]), NE+NNLS (Neighbor Embedding with Non-Negative Least Squares, similar to Bevilacqua *et al.* [10]), GR (Global Regression) and ANR (Anchored Neighborhood Regression) of Timofte *et al.* [1]; the sparse coding method of Yang *et al.* [16], the efficient sparse coding method of Zeyde *et al.* [11], and the Simple Functions (SF) method of Yang and Yang [12]. In addition we briefly report to the very recent Convolutional Neural Network method (SR-CNN) of Dong *et al.* [18] which uses the same benchmark as us (training data, Set5, Set14, as in [1]).

[1] All the codes are publicly available at: http://www.vision.ee.ethz.ch/~timofter/

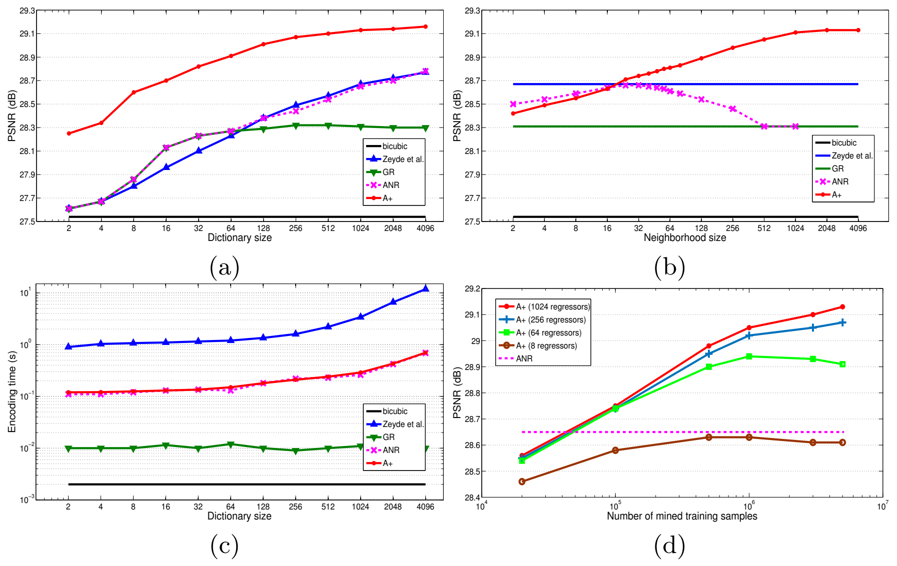


Fig. 2. Parameters influence on performance on average on Set14. (a) Dictionary size for *A+*, *ANR*, *GR*, and *Zeyde* et al. versus PSNR; (b) Neighborhood size for *A+* and *ANR* versus PSNR, with dictionary size fixed to 1024; *GR*, *Zeyde* et al. and *Bicubic* interpolation are provided for reference. (c) Dictionary size for *A+*, *ANR*, *GR*, and *Zeyde* et al. versus encoding time; *Bicubic* is for reference. (d) Number of mined training samples versus PSNR for *A+* with different number of regressors (aka dictionary size); *ANR* with a dictionary of 1024 is provided for reference.

Features We use the same LR patch features as Zeyde *et al.* [11] and Timofte *et al.* [1]. Therefore, we refer the reader to their original works for a discussion about features.

4.2 Parameters

In this subsection we analyze the main parameters of our proposed method, and at the same time compare on similar settings with other methods, especially with ANR since it is the closest related method. The standard settings we use are upscaling factor $\times 3$, 5000000 training samples of LR and HR patches, a dictionary size of 1024, and a neighborhood size of 2048 training samples for A+ and 40 atoms for ANR, or the best or common parameters for the other methods as reported in their respective original works. Fig. 2 depicts the most relevant results of the parameter settings.

Dictionaries There is a whole discussion concerning what are the best dictionaries of pairs of LR and HR patches (or pair samples hereafter). Some NE methods prefer to keep all the training samples, leaving the task of sublinear retrieval of relevant neighboring samples to a well-chosen data structuring. The sparse coding methods, the recent NE methods (SF, ANR) and the parameter study of Timofte *et al.* [1] argue that learning smaller dictionaries is beneficial, greatly reducing the search complexity and even improving in speed or quality. We adhere to the setup from [1] and we vary the size of the learned dictionary from 2 up to 4096 atoms. The training pool of samples extracted from the same training images as used also in [1] or [16] is in the order of 0.5 million. The results are depicted in the Fig. 2 (a) for quantitative Peak Signal-to-Noise Ratios (PSNRs) and (c) for encoding time. We refer the reader to [1] for a study about NE+LS, NE+LLE and NE+NNLS. We focus mainly on ANR (with a neighborhood size of 40 atoms) and A+ (with a neighborhood size of 2048 samples), and provide results also for bicubic interpolation, Zeyde *et al.* and GR as comparison. Both A+ and ANR have the same time complexity, the running times are almost identical for the same learned dictionary size. The difference is in the quantitative PSNR performance, where A+ is clearly ahead of ANR. In fact A+ is ahead of all compared methods regardless of the dictionary size.

Neighborhoods The size of the neighborhood that is used by the neighboring embedding methods to compute the regression of the LR input patch in order to reconstruct the HR output patch is usually a critical parameter of most NE methods. Timofte *et al.* [1] already show how sensitive this parameter is for NE+LS, NE+NNLS or NE+LLE. In the experiment from Fig. 2 (b) we compare the behavior of A+ and ANR over the same dictionary, while varying the neighborhood size. Note that while A+ forms the neighborhoods from the closest training LR patches in Euclidean distance, ANR builds the local neighborhoods from the other atoms in the learned dictionary using the correlation as similarity measure. As we see, ANR peaks at a neighborhood size of 40, while our A+ faces a plateau above 1024. We had 5 million samples in the training pool, and this is a possible explanation why A+ with 1024 atoms faces a plateau above 1024 neighborhood sizes, as we will investigate in the next subsection. From now on, unless mentioned otherwise, A+ will always use neighborhoods of size 2048 for each of its atoms.

Training samples and dictionary size In the previous subsection we found that A+ seems to face a plateau in relation with the available pool of training samples from where it can pick its neighborhoods. In Fig. 2 (d) we present the results of an experiment showing the relation between the amount of training samples mined from the training set of images and the dictionary size of our method. Since for each atom we learn one regressor from its neighborhood, we decide to evaluate A+ with 8, 64, 256, and 1024 regressors respectively (or in other words, its dictionary size or number of anchoring points/atoms). For reference we also plot the ANR result with a 1024 dictionary and its optimal neighborhood size of 40 atoms. The result of the experiment is relevant in that we can see that the larger we make the training pool, the better the performance of the regressors becomes, even if the number of regressors is very small. This is due to the fact that by having a larger training pool of samples, the density of

Table 1. PSNR and running time for upscaling factor ×3 for Set14. All methods are trained on the same images and, with the exception of Yang *et al.* (1022 atoms), share a dictionary of 1024 atoms; A+ trained with 5000000 samples. A+ and ANR are 5 times faster than Zeyde *et al.* If we consider only the encoding time, our A+ takes 0.23s on average, being 14 times faster than Zeyde *et al.*, and 10 times faster than NE+LS.

Set14	Bicubic		Yang *et al.* [16]		Zeyde *et al.* [11]		GR [1]		ANR [1]		NE+LS		NE+NNLS		NE+LLE		**A+**	
images	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
baboon	23.2	0.0	23.5	138.7	23.5	4.1	23.5	0.5	**23.6**	0.7	23.5	2.8	23.5	28.1	23.6	5.4	**23.6**	0.7
barbara	26.2	0.0	26.4	146.6	**26.8**	7.4	**26.8**	1.0	26.7	1.4	26.7	5.3	26.7	48.7	26.7	9.4	26.5	1.3
bridge	24.4	0.0	24.8	169.5	25.0	4.3	24.9	0.5	25.0	0.8	24.9	3.1	24.9	30.2	25.0	5.8	**25.2**	0.8
coastguard	26.6	0.0	27.0	39.9	27.1	1.7	27.0	0.2	27.1	0.3	27.0	1.1	27.0	12.0	27.1	2.2	**27.3**	0.3
comic	23.1	0.0	23.9	59.8	24.0	1.5	23.8	0.2	24.0	0.3	23.9	1.0	23.8	10.3	24.0	1.9	**24.4**	0.3
face	32.8	0.0	33.1	22.6	33.5	1.3	33.5	0.1	33.6	0.2	33.5	0.9	33.5	8.4	33.6	1.6	**33.8**	0.2
flowers	27.2	0.0	28.2	88.6	28.4	3.1	28.1	0.4	28.5	0.5	28.3	2.1	28.2	21.4	28.4	4.0	**29.0**	0.6
foreman	31.2	0.0	32.0	30.4	33.2	1.7	32.3	0.2	33.2	0.3	33.2	1.2	32.9	11.5	33.2	2.1	**34.3**	0.3
lenna	31.7	0.0	32.6	76.7	33.0	4.4	32.6	0.6	33.1	0.8	33.0	3.0	32.8	30.5	33.0	5.8	**33.5**	0.8
man	27.0	0.0	27.8	120.9	27.9	4.2	27.6	0.5	27.9	0.8	27.9	3.0	27.7	28.9	27.9	5.9	**28.3**	0.8
monarch	29.4	0.0	30.7	128.0	31.1	6.5	30.4	0.8	31.1	1.2	30.9	4.7	30.8	46.1	30.9	8.6	**32.1**	1.2
pepper	32.4	0.0	33.3	78.9	34.1	4.3	33.2	0.5	33.8	0.7	33.9	2.9	33.6	28.8	33.8	5.8	**34.7**	0.8
ppt3	23.7	0.0	25.0	106.6	25.2	5.4	24.6	0.7	25.0	1.1	25.1	4.0	24.8	35.3	24.9	7.6	**26.1**	1.0
zebra	26.6	0.0	28.0	122.9	28.5	3.7	27.9	0.4	28.4	0.7	28.3	2.6	28.1	25.7	28.3	4.9	**29.0**	0.7
average	27.54	0.01	28.31	95.00	28.67	3.83	28.31	0.47	28.65	0.69	28.59	2.68	28.44	26.15	28.60	5.08	**29.13**	**0.69**
avg.encoding		0.01		∼90.00		3.37		0.01		0.23		2.22		25.69		4.62		**0.23**

points favorably placed near the anchoring atoms increases and the regression can better fit a manifold in the proximity of the unit hypersphere of the atoms. Nevertheless, for better performance the neighborhood size (here fixed at 2048) should be adjusted as well to the number of mined training samples. And this especially when the number of training samples is much larger or smaller than the number of regressors times the neighborhood size. Our assumption of the space spanned by anchored neighborhoods on atoms on the unit hypersphere seems to hold.

Our experiment, which we stopped after harvesting 5 million training samples, shows surprisingly that we are able to learn sufficiently accurate as low as 8 regressors to the high resolution space to get close to the 1024 regressors used in the ANR method, but algorithmically we are up to 128× faster! Also, we reach 29.13 dB on Set14 with 1024 regressors and 5 million extracted samples (with 0.5 million samples we get only 28.97 dB).

Noteworthy is that the performance of A+ does not seem to saturate with the number of extracted samples, which allows better performance at the price of increasing the training time which is only about 15 minutes for 1024 regressors and 5 millions extracted patches on our tested setup (Intel i7-4770K with 16 GBytes of RAM).

4.3 Performance

In order to assess the quality of our proposed A+ method, we tested it on 3 datasets (Set5, Set14, B100) for 3 upscaling factors (×2, ×3, ×4). We report quantitative PSNR results, as well as running times for our bank of methods run under the same testing conditions. In Table 3 we summarize the quantitative results, while in Fig. 3, 4, and 5 we provide a visual assessment on three images. [2]

[2] For more results: http://www.vision.ee.ethz.ch/∼timofter/

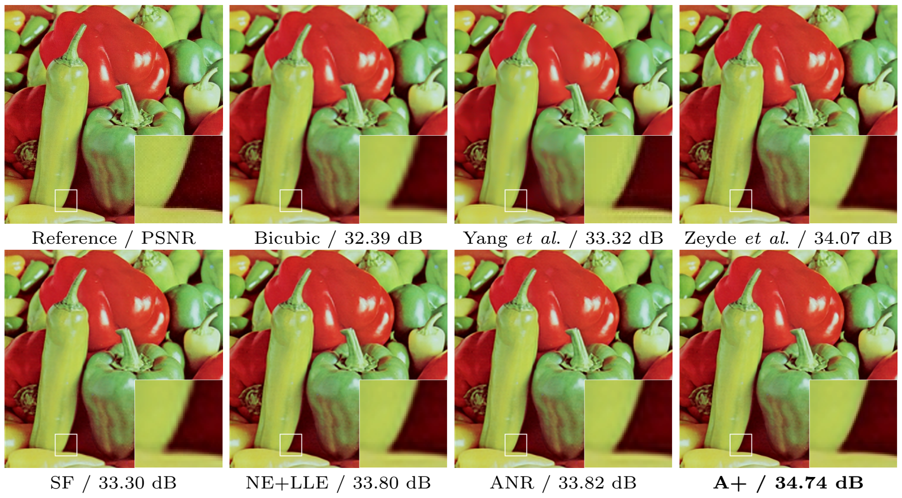


Fig. 3. 'Pepper' image from Set14 with upscaling ×3.

Quality Timofte *et al.* [1] show that most of the current neighbor embedding methods are able to reach comparable performance quality for the right sets of parameters, such as dictionary size, training choices, features, and internal regulatory parameters. The critical difference among most methods (including neighborhood embedding, sparse coding methods, and not only these) is the time complexity and the running time during testing (and training). In Table 1 we evaluate on the same trained dictionary, with their best parameters, a set of methods (as proposed in [1]) on Set14 dataset. Yang *et al.* [16] uses a different dictionary, but of comparable size (1022 vs 1024) and learned on the same training images. In Table 1 we show results for upscaling factor ×3, and in Table 2 we report results for ×2, ×3, and ×4 upscaling factors. As repeatedly shown, our proposed method is the best method quality-wise, improving on average 0.26dB (B100,×4) up to 0.72dB (Set5,×2) over the next top methods, Zeyde *et al.* or ANR. The very recent SRCNN method [18] is 0.2dB behind A+ on Set5 and more than 0.13dB on Set14 according to its published results. At the same time, A+ is very efficient, has the time complexity of ANR and except for bicubic interpolation and the Global Regression (GR) method [1], which are clearly outperformed in quality, A+ is the fastest method for a given target in quality of the SR result (see Fig. 2 (a) and (c) for PSNR and time vs. dictionary size for different methods). In Fig. 3, 4, and 5 we show how A+ has a visual quality comparable or superior to the other compared methods on a couple of images.

Running Time The time complexity of A+ for encoding LR input patches to HR output patches is linear in the number of input image patches and linear in the number of anchoring atoms. One can easily get sub-linear time complexity in the number of atoms by using any popular search structure, since we just need to retrieve the closest anchor for a specific patch, followed by a (fixed time cost) projection to the HR patch. ANR shares the time complexity (see

Table 2. PSNR and running time for upscaling factors ×2, ×3 and ×4 for Set5. All methods are trained on the same images and share a dictionary of 1024 atoms; A+ trained with 5000000 samples. For upscaling factor 3, A+ and ANR are 5 times faster than Zeyde *et al.* 120 times faster than Yang *et al.* and 4 times faster than NE+LS.

Set5		Bicubic		Yang *et al.* [16]		Zeyde *et al.* [11]		GR [1]		ANR [1]		NE+LS		NE+NNLS		NE+LLE		**A+**	
images	*Scale*	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	**PSNR**	**Time**
baby	x2	37.1	0.0	–	–	38.2	9.9	38.3	0.8	38.4	1.3	38.1	6.5	38.0	71.1	38.3	12.9	**38.5**	1.3
bird	x2	36.8	0.0	–	–	39.9	3.0	39.0	0.2	40.0	0.4	39.9	2.0	39.4	22.0	40.0	3.9	**41.1**	0.4
butterfly	x2	27.4	0.0	–	–	30.6	2.4	29.1	0.2	30.5	0.3	30.4	1.7	30.0	17.9	30.4	3.2	**32.0**	0.3
head	x2	34.9	0.0	–	–	35.6	2.9	35.6	0.2	35.7	0.4	35.5	1.9	35.5	20.8	35.6	3.8	**35.8**	0.4
woman	x2	32.1	0.0	–	–	34.5	2.9	33.7	0.2	34.5	0.4	34.3	2.1	34.2	21.0	34.5	3.8	**35.3**	0.4
average	x2	33.66	0.00	–	–	35.78	4.25	35.13	0.33	35.83	0.54	35.66	2.83	35.43	30.56	35.77	5.51	**36.55**	0.55
baby	x3	33.9	0.0	34.3	89.6	35.1	4.3	34.9	0.6	35.1	0.8	35.0	3.1	34.8	29.9	35.1	5.8	**35.2**	0.8
bird	x3	32.6	0.0	34.1	35.4	34.6	1.3	33.9	0.2	34.6	0.3	34.4	0.9	34.3	9.1	34.6	1.8	**35.5**	0.2
butterfly	x3	24.0	0.0	25.6	32.9	25.9	1.1	25.0	0.1	25.9	0.2	25.8	0.7	25.6	7.0	25.8	1.4	**27.2**	0.2
head	x3	32.9	0.0	33.2	25.3	33.6	1.3	33.5	0.2	33.6	0.2	33.5	0.9	33.5	8.4	33.6	1.7	**33.8**	0.2
woman	x3	28.6	0.0	29.9	31.1	30.4	1.3	29.7	0.2	30.3	0.2	30.2	0.9	29.9	8.4	30.2	1.7	**31.2**	0.2
average	x3	30.39	0.00	31.42	42.86	31.90	1.86	31.41	0.24	31.92	0.34	31.78	1.29	31.60	12.56	31.84	2.46	**32.59**	0.35
baby	x4	31.8	0.0	–	–	33.1	2.7	32.8	0.4	33.0	0.6	32.9	1.9	32.8	16.5	33.0	3.5	**33.3**	0.6
bird	x4	30.2	0.0	–	–	31.7	0.8	31.3	0.1	31.8	0.2	31.6	0.6	31.5	4.9	31.7	1.0	**32.5**	0.2
butterfly	x4	22.1	0.0	–	–	23.6	0.6	23.1	0.1	23.5	0.1	23.4	0.4	23.3	3.8	23.4	0.8	**24.4**	0.1
head	x4	31.6	0.0	–	–	32.2	0.7	32.1	0.1	32.3	0.2	32.2	0.5	32.1	4.7	32.2	1.0	**32.5**	0.2
woman	x4	26.5	0.0	–	–	27.9	0.7	27.4	0.1	27.8	0.2	27.6	0.5	27.6	4.6	27.7	1.0	**28.6**	0.2
average	x4	28.42	0.00	–	–	29.69	1.12	29.34	0.17	29.69	0.25	29.55	0.78	29.47	6.89	29.61	1.45	**30.28**	0.24

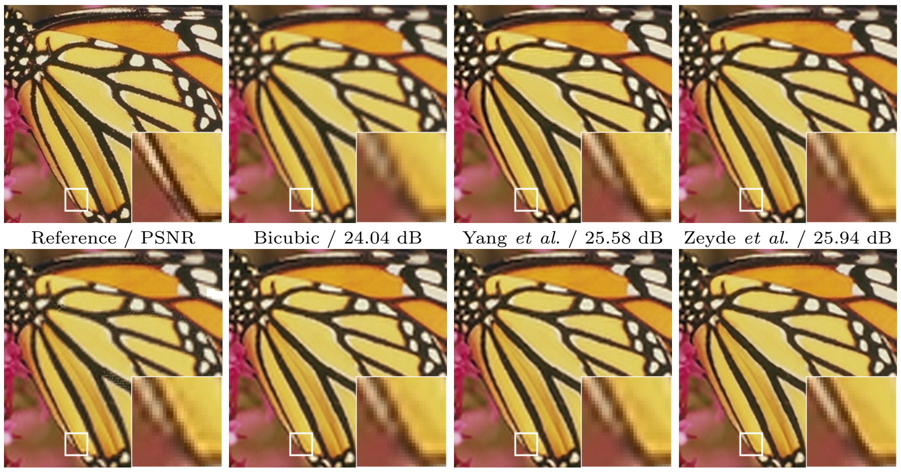
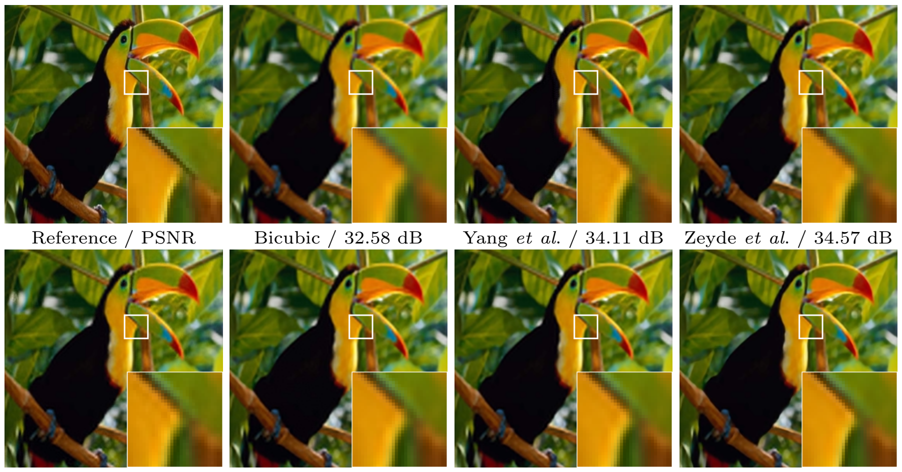

Reference / PSNR
Bicubic / 24.04 dB
Yang *et al.* / 25.58 dB
Zeyde *et al.* / 25.94 dB
SF / 24.40 dB
NE+LLE / 25.75 dB
ANR / 25.90 dB
A+ / 27.24 dB

Fig. 4. 'Butterfly' image from Set5 with upscaling ×3.

Table 3. PSNR and running time for upscaling factors ×2, ×3 and ×4 for Set14, Set5, and B100. All methods are trained on the same images and share a dictionary of 1024 atoms, except for SF, which is trained with 1024 clusters and corresponding functions.

		Bicubic		SF [12]		Zeyde *et al.* [11]		GR [1]		ANR [1]		NE+LS		NE+NNLS		NE+LLE		**A+**	
Dataset	*Scale*	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	**PSNR**	**Time**
	x2	33.66	0.00	35.63	20.46	35.78	4.25	35.13	0.33	35.83	0.54	35.66	2.83	35.43	30.56	35.77	5.51	**36.55**	0.55
Set5	x3	30.39	0.00	31.27	11.89	31.90	1.86	31.41	0.24	31.92	0.34	31.78	1.29	31.60	12.56	31.84	2.46	**32.59**	0.35
	x4	28.42	0.00	28.94	6.42	29.69	1.12	29.34	0.17	29.69	0.25	29.55	0.78	29.47	6.89	29.61	1.45	**30.29**	0.24
	x2	30.23	0.00	31.04	39.11	31.81	8.58	31.36	0.73	31.80	1.15	31.69	5.69	31.55	60.53	31.76	11.27	**32.28**	1.20
Set14	x3	27.54	0.00	28.20	24.59	28.67	3.83	28.31	0.47	28.65	0.69	28.59	2.68	28.44	26.15	28.60	5.08	**29.13**	0.69
	x4	26.00	0.00	26.25	11.99	26.88	2.42	26.60	0.42	26.85	0.57	26.81	1.68	26.72	14.49	26.81	3.07	**27.33**	0.56
	x2	29.32	0.00	30.35	10.15	30.40	5.80	30.23	0.45	30.44	0.73	30.36	3.79	30.27	39.83	30.41	7.50	**30.78**	0.76
B100	x3	27.15	0.00	27.76	4.94	27.87	2.54	27.70	0.31	27.89	0.45	27.83	1.81	27.73	17.57	27.85	3.47	**28.18**	0.46
	x4	25.92	0.00	26.19	2.75	26.51	1.53	26.37	0.25	26.51	0.35	26.45	1.09	26.41	2.04	26.47	2.01	**26.77**	0.35



Reference / PSNR — Bicubic / 32.58 dB — Yang *et al.* / 34.11 dB — Zeyde *et al.* / 34.57 dB

SF / 33.98 dB — NE+LLE / 34.56 dB — ANR / 34.60 dB — **A+ / 35.54 dB**

Fig. 5. 'Bird' image from Set5 with upscaling $\times 3$.

Fig. 2), but requires much larger dictionaries of anchoring points for comparable performance, which makes it considerably slower as shown in our experiments. A+ is order(s) of magnitude faster than another successful efficient sparse coding approach, Zeyde *et al.* [11]. At the same time, A+ is 0.2dB up to 0.7dB better at any given dictionary size. With as few as 16 atoms A+ outperforms ANR and Zeyde *et al.* with dictionaries of 2048 atoms. This corresponds to an algorithmic speed up of 128 over ANR for the same quality level.

5 Conclusions

We proposed an enhanced highly efficient example-based super-resolution method, which we named Adjusted Anchored Neighborhood Regression, or shortly – A+. A+ succeeds in substantially surpassing the shortcomings of its predecessors such as ANR or SF. We proposed a different interpretation of the LR space, as a joint space of subspaces spanned by anchoring points and their closed neighborhood of prior samples. While the anchoring points are the unit l_2-norm atoms of a sparse dictionary, the characterizing neighborhood is formed by mined samples from the training samples. For each such atom and neighborhood a regression is learned offline and at test time this is applied to the correlated low resolution samples to super-resolve it. A+ is shown on standard benchmarks to improve 0.2dB up to 0.7dB in performance over state-of-the-art methods such as ANR or SF. At the same time it is indisputably the fastest method. It has the lowest time complexity and uses orders of magnitude less anchor points than ANR or SF for substantially better performance. As future work we plan to explore A+ for video processing and other real-time critical applications.

Acknowledgement. This work was partly supported by the ETH General Founding (OK) and the Flemish iMinds framework.

References

1. Timofte, R., De Smet, V., Van Gool, L.: Anchored Neighborhood Regression for Fast Example-Based Super Resolution. ICCV (2013) 1920–1927
2. Sun, J., Xu, Z., Shum, H.-Y.: Image super-resolution using gradient profile prior. CVPR (2008) 1–8
3. Glasner, D., Bagon, S., Irani, M.: Super-Resolution from a Single Image. ICCV (2009) 349–356
4. Yang, J., Wright, J., Huang, T. S., Ma Y.: Image super-resolution as sparse representation of raw image patches. CVPR (2008) 1–8
5. Sun, J., Zhu, J., Tappen, M. F.: Context-constrained hallucination for image super-resolution. CVPR (2010) 231–238
6. Michaeli, T., Irani, M.: Nonparametric Blind Super-resolution. ICCV (2013) 945–952
7. Efrat, N., Glasner, D., Apartsin, A., Nadler, B., Levin, A.: Accurate Blur Models vs. Image Priors in Single Image Super-Resolution. ICCV (2013) 2832–2839
8. Freeman, W. T., Pasztor, E. C., Carmichael, O. T.: Learning low-level vision. IJCV **40(1)** (2000) 25–47
9. Chang, H., Yeung, D.-Y., Xiong, Y.: Super-Resolution through Neighbor Embedding. CVPR (2004) 275–282
10. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi Morel, M.-L.: Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding. BMVC (2012) 1–10
11. Zeyde, R., Elad, M., Protter, M.: On Single Image Scale-Up Using Sparse-Representations. Curves and Surfaces (2012) 711–730
12. Yang, C.-Y., Yang, M.-H.: Fast Direct Super-Resolution by Simple Functions. ICCV (2013) 561–568
13. Roweis, S., Lawrence, S.: Nonlinear dimensionality reduction by locally linear embedding. Science (2000) 2323–2326
14. Timofte, R., Van Gool, L.: Adaptive and weighted collaborative representations for image classification. Pattern Recognition Letters **43** (2014) 127–135
15. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. ICCV (2001) 416–423
16. Yang, J., Wright, J., Huang, T. S., Ma, Y.: Image Super-Resolution Via Sparse Representation. IEEE Trans. on Image Processing **19(11)** (2010) 2861–2873
17. De Smet, V., Namboodiri, V. P., Van Gool, L. J.: Nonuniform image patch exemplars for low level vision. WACV (2013) 23–30
18. Dong, C., Loy, C. C., He, K., Tang, X.: Learning a Deep Convolutional Network for Image Super-Resolution. ECCV (2014) 184–199