Naive Bayes Super-Resolution Forest

Jordi Salvador Eduardo Pérez-Pellitero
Technicolor R&I Hannover
{jordi.salvador,eduardo.perezpellitero}@technicolor.com

Abstract

This paper presents a fast, high-performance method for super resolution with external learning. The first contribution leading to the excellent performance is a bimodal tree for clustering, which successfully exploits the antipodal invariance of the coarse-to-high-res mapping of natural image patches and provides scalability to finer partitions of the underlying coarse patch space. During training an ensemble of such bimodal trees is computed, providing different linearizations of the mapping. The second and main contribution is a fast inference algorithm, which selects the most suitable mapping function within the tree ensemble for each patch by adopting a Local Naive Bayes formulation. The experimental validation shows promising scalability properties that reflect the suitability of the proposed model, which may also be generalized to other tasks. The resulting method is beyond one order of magnitude faster and performs objectively and subjectively better than the current state of the art.

1. Introduction

Ongoing research on example-based super resolution has recently provided large performance improvements in terms of accuracy and efficiency thanks to the introduction of the latest machine learning approaches. One possible strategy is internal learning [10, 8, 29], where the cross-scale self-similarity property of small (square) natural image patches is exploited to regress the appearance of high-resolution patches. Despite the nice properties of these methods, e.g. their implicit adaptivity to the image contents, an important practical drawback is the computational cost induced by the required nearest-neighbor search.

In order to alleviate this problem, one possibility is to exploit the latest and efficient approximate nearest-neighbor search algorithms [2, 13, 11, 22, 25]. The alternative is the progressive reduction of the search range for similar patches presented by the latest best performing internal learning methods [8, 29]. The evolution from the work of Glasner et al. [10] to that of Yang et al. [29] is remarkable: The search on the entire multi-scale pyramid of each intermediate image towards the desired scale in the former is reduced to an iterated in-place example selection (one of just 9 possible patches in the immediately lower scale) in the latter thanks to the incorporation of additional prior knowledge from offline training. The exploitation of this additional prior also requires a search, which is common to most external learning approaches as we observe below. An additional limitation in internal learning methods exploiting cross-scale self-similarity is that the processing must be iteratively applied in small scaling factors to fullfill the self-similarity assumption, which translates into a costlier processing.

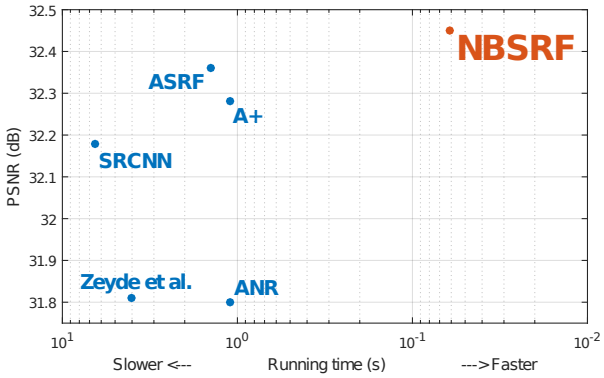

Figure 1. The proposed Naive Bayes Super-Resolution Forest clearly improves PSNR and runtime in comparison to the state of the art. (SRCNN uses the public slower implementation.)

In contrast, recent external learning approaches [23, 19, 24, 20] cast the super-resolution problem as a mapping from either low-res or coarse (upscaled by interpolation) patches to high-res ones in a single step. Since the mapping function is non-linear, it is beneficious to properly split it into locally linear mapping functions. During an offline training stage, low-res or coarse patches can be clustered (e.g. extracting centroids with K-SVD [1]) under the assumption of mapping function linearity close to each centroid. This approach, which contrasts with former efforts in costly sparse coding-based regression [30, 31, 18], provides a streamlined online inference: (1) determine the best cluster (linearization) for each patch; (2) apply the precomputed linear mapping function; and (3) reconstruct by patch overlapping.

Contributions. We present a novel example-based super-resolution method with external learning that provides a fast, scalable and accurate solution by tackling the most demanding problem in the inference stage (i.e. the selection of the local linearization for each patch) and the corresponding training process. In more detail, the contributions of this paper are: **(1)** a hierarchical manifold learning strategy with *bimodal* trees that allows grouping antipodal patches and provides fast local linearization search (Section 3); **(2)** an efficient Local Naive Bayes strategy for per-patch tree selection (Section 4); and **(3)** the resulting method, namely *Naive Bayes Super-Resolution Forest* (NBSRF) that amply improves speed and both objective and perceived quality in comparison to the state of the art, as Fig. 1 reflects.

2. Related work

The super-resolution problem is that of estimating a high-resolution version X of a low-resolution observed image Y generated by the model

$$Y = (X * H_d) \downarrow s, \tag{1}$$

where $*$ denotes image convolution, H_d is a filter preventing aliasing from appearing in Y and $\downarrow s$ is a downsampling operator with scaling factor s (i.e. the desired upscaling factor). A common asumption in the example-based super-resolution literature is that H_d does not introduce additional blur in Y, so the latter is assumed to be sharp. In [17] a method is proposed, based on internal learning, to estimate the overall blur kernel for super-resolving blurred images, but in the following we keep the no-blur assumption to be consistent with most recent methods. Both images X and Y are decomposed into overlapping patches $\{x\}$ and $\{y\}$.

2.1. External learning

Yang et al. [30] proposed introducing additional knowledge about the scaling process by means of Sparse Coding. During the training stage, a large set of patches extracted from natural images and their downscaled (low-res) counterparts are used to generate two coupled sparse dictionaries, one for high-res D_h and another one for low-res D_l. The inference is based on a decomposition of each low-res patch y as a sparse linear combination of the atoms in D_l, which results in a costly optimization process for each patch that can be expressed as

$$\min_{\alpha} \|y - D_l\alpha\|_2^2 + \lambda\|\alpha\|_0. \tag{2}$$

The data term enforces similarity between the observed patch and the reconstruction, whereas the sparse regularization enforces a small number of non-zero entries in the decomposition α. The reconstruction of the high-res patch x is straight-forward once α is known: $x = D_h\alpha$, and the full image reconstruction can be accomplished by overlapping all reconstructed patches.

A major improvement on both speed and accuracy was obtained by [31] with the introduction of K-SVD [1] to train a coarse dictionary (for coarse patches only, without involving the high-res ones in the optimization) and Orthogonal Matching Pursuit (OMP) [26] to solve the decomposition problem (similar to the one in Eq. 2) during the inference stage. Despite the improvements, the use of OMP during inference is clearly the bottleneck. Consequently, Anchored Neighbor Regression (ANR) [23] proposed to relax the sparsity constraint in Eq. 2 and anchor the set of most similar atoms, resulting in a L_2-regularized regression that can be computed in closed form during training:

$$\min_{\alpha} \|y - N_l(y)\alpha\|_2^2 + \lambda\|\alpha\|_2^2, \tag{3}$$

where $N_l(y)$ contains just the nearest neighbors within D_l of one of the atoms. The resulting linear mapping for the chosen atom in the dictionary is $N_h(y)\alpha$, where $N_h(y)$ is the high-res counterpart to $N_l(y)$ and the inference stage is reduced to a relatively costly exhaustive nearest-neighbor search, a matrix multiplication (locally linear mapping function) and reconstruction by overlapping.

The Simple Functions approach of Yang and Yang [28] proposes an alternative model where clustering is applied directly onto low-res patches and then a separate mapping from low-res to high-res patches is learned. Their simple mapping function of choice is an affine transformation that can be learned offline for each cluster. Again, a practical limitation of the method is that the affine mapping for each patch is determined by an exhaustive search over all possible clusters.

More recently, [19, 24] have proposed to obtain a dictionary trained with K-SVD like in [23] with the difference that, instead of learning the linear regression function of each atom in the dictionary using only other atoms, the entire set of training samples are used. This produces a better sampling of the coarse manifold and also provides a unification of [28] and [23].

2.2. Locally linear map search

As hinted above, perhaps the most affecting practical bottleneck in many super-resolution approaches relying on local linearizations of the mapping function is the exhaustive search stage [23, 28, 24]. One possible solution is presented in [19], where a fast nearest-neighbor search is built such that the clustering induced by the K-SVD dictionary atoms can be efficiently exploited in the inference stage, resulting in large speed-ups. The idea is to construct this fast search strategy based on Spherical Hashing [11] to get logarithmic-time access to the anchor points (and their corresponding linear mapping functions) obtained by K-SVD. However, whereas K-SVD usually leads to a good selection

of the most relevant dictionary atoms for reconstructing all possible observed data, it also involves a rather slow process that might be impractical for large training datasets.

An alternative and more uniform solution is to perform unsupervised hierarchical clustering, e.g. [9, 16], in order to both determine clusters of similar patches (or features) and provide an intrinisic logarithmic-time search during the inference stage. This strategy is not new to super-resolution as the random projections approach [9] (which approximates [16] without computing the PCA of the elements in each hierarchical cluster) has already been employed for in-place regression [29]. Very recently, [20, 12] have also proposed to combine regressions from random tree ensembles to improve the accuracy. Since hierarchical approaches are capable of solving both the clustering and efficient inference problems, in the following we explore the applicability on our fast and accurate super-resolution framework.

3. Hierarchical manifold learning

Our model is in essence related to [28], in the sense that we aim at providing a direct mapping from coarse to high-res patches by dividing the input coarse space into clusters and then learning a locally linear mapping to the high-res patches, and also to [24, 19], so that our mapping is actually a *correction* layer for an initial coarse estimation of the high-res patch. The coarse estimate of X is thus first obtained via iterative back-projection (IBP) from the low-res input Y

$$\tilde{X}^{(n+1)} := \tilde{X}^{(n)} + H_u * ((Y - (\tilde{X}^{(n)} * H_d) \downarrow s) \uparrow s), \quad (4)$$

where the superscript (n) indicates the iteration and H_u is an interpolation filter. Typically, a small number of iterations, e.g. 2 or 3, is sufficient to reach convergence. At a patch level, the reconstruction model continues as

$$x = \tilde{x} + \mathcal{R}(\tilde{x} - \bar{x}) = \tilde{x} + \mathcal{R}(\tilde{x}_0), \quad (5)$$

where $\mathcal{R}$ is a nonlinear regression function and $\tilde{x}_0$ is the mean-subtracted (or floating) version of $\tilde{x}$. Whereas other approaches (e.g. [31, 23, 24, 19]) use handcrafted features as input to the regression problem, below we explain how features can be adaptively computed in order to select one of the M available local linearizations R_i of $\mathcal{R}$ obtained during training.

We further observe that given a patch x with a certain structure, the scaled version αx contains the same structure with different contrast. Since this argument is valid for both x and $\tilde{x}_0$ when they are related by a linear transform, it makes sense to group all patches with the same structure by normalizing them ($\hat{x} = \tilde{x}_0 \|\tilde{x}_0\|_2^{-1}$). The result is that all possible patch structures lie on a unitary hypersphere centered at the origin akin to the toy example in Fig. 2 top.

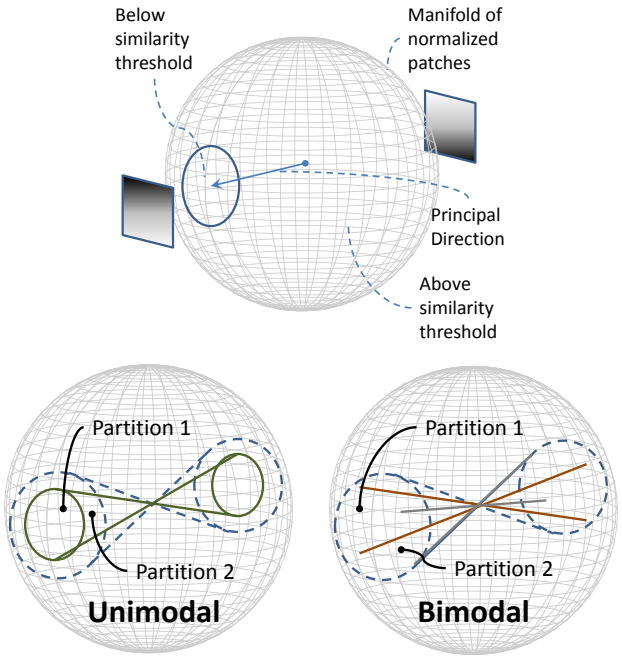

Figure 2. Top, typical unimodal partitions on a normalized-feature space. The spherical cap (delimited by the solid line) is the fraction of the manifold that can be described with a single principal direction. Bottom, clustering for antipodal patch grouping with unimodal and bimodal partitions. Note the more balanced partitions of the (colored) data possible with the bimodal scheme.

Based on the good results in [29] for in-place example regression one might consider a similar space partitioning strategy, i.e. using *unimodal* partition trees: for each node, split data based on the thresholding of a feature (the response to a certain filter). This is the mechanism underlying the PCA tree [16], its random projection approximation [9] and also the faster k-D tree. In the latter, a set of features is precomputed for all data and the spliting is based on the thresholding of the most sensitive feature for each node, whereas the PCA tree and its approximation provide an adaptive computation of relevant features during the root-to-leaf tree traversal.

This strategy is valid when data are distributed as a thin manifold [9]. However, inspecting the configuration of our problem, depicted in Fig. 2 top (with all data lying on a hypersphere), we observe that partitioning data in this fashion would be rather suboptimal. The resulting partition with unimodal approaches is rather unbalanced in our setup: the set of data lying out of the inclusion partition (projection above threshold) is much more heterogeneus than the one lying inside (below threshold).

3.1. Antipodal symmetry

In addition to the normalization of data, it is also important to observe that antipodal points, i.e. $\hat{x}$ and $-\hat{x}$, are also

representations of the same structure with a scaling factor. An example of such pair of points is also shown in Fig. 2 top. Hence, an efficient hierarchical partition of the space in our problem should aim to also keep all antipodal pairs within the same cluster. In order to do so, the optimal metric is the Absolute Value of the Cosine Similarity (AVCS), $|\hat{a} \cdot \hat{b}|$, where $\hat{a}$ and $\hat{b}$ are two data on the unitary hypersphere. In Fig. 2 bottom we illustrate the potentially better balanced bimodal partitions when combined with the AVCS metric. Note that the unimodal partitions may yield a rather compact centered cluster and a surrounding one with high variance vs. the more uniform bimodal partitions.

3.2. Bimodal regression tree training

During training we build our partition tree such that, for each node, the splitting criterion is able to discriminate the two most relevant clusters grouping antipodal data. Given the AVCS metric, using linear strategies to discriminate clusters, e.g. PCA, is not suitable (Fig. 2 bottom left). We therefore need to rely on subspace clustering techniques.

For simplicity and efficiency, we adopt a modification of k-means [14], which can be interpreted as an antipodally invariant spherical k-means algorithm[1]. The modifications of the two iterative stages of the algorithm for k classes $\mathcal{C}_i, 1 \leq i \leq k$ result in the iteration

E-step. $\mathcal{C}_i^{(n)} = \{\hat{x} : |\hat{x} \cdot \mu_i^{(n)}| \geq |\hat{x} \cdot \mu_j^{(n)}| \forall j, 1 \leq j \leq k\}$

M-step. $\mu_i^{(n+1)} = \| \sum_{\hat{x} \in \mathcal{C}_i^{(n)}} \sigma \hat{x} \|_2^{-1} \sum_{\hat{x} \in \mathcal{C}_i^{(n)}} \sigma \hat{x}$,

where $\sigma = \text{sign}(\hat{x} \cdot \mu_i^{(n)})$. The resulting partition criterion for a node in the *bimodal* tree is $\text{argmax}_i |\mu_i \cdot \hat{x}|, i = 1, 2$. Note that this is fundamentally different from the criteria found in unimodal trees, where only one projection takes place. In Section 5 we discuss the advantages of using the bimodal tree for space partitioning.

Locally linear regressors. Since during training we have access to coarse patches and their high-res counterparts (low-res images are generated by downscaling the set of training images as in Eq. 1), all $\{\tilde{x}_0\}$, $\{\tilde{x}\}$ and $\{x\}$ corresponding to all $\{\hat{x}\}$ lying in one of the M clusters (tree leaves) are used to compute the local linearization of $\mathcal{R}$ in the form of the regression matrix $R_i, 1 \leq i \leq M$. Let $\tilde{\mathbf{X}}_0$, $\tilde{\mathbf{X}}$ and $\mathbf{X}$ be the matrices obtained by stacking all these (vectorized) patches as columns. The regression matrix is obtained from Eq. 5 as

$$R_i = (1 + \lambda)(\mathbf{X} - \tilde{\mathbf{X}})\tilde{\mathbf{X}}_0^\top (\tilde{\mathbf{X}}_0 \tilde{\mathbf{X}}_0^\top + \lambda \mathbf{I})^{-1}, \quad (6)$$

where λ is a small regularization factor, e.g. 10^{-4}, to avoid numerical stability issues.

[1] In our experiments with Forgy (random) initialization this choice provided PSNR performance similar and in some cases even better than the much costlier K-SVD [1].

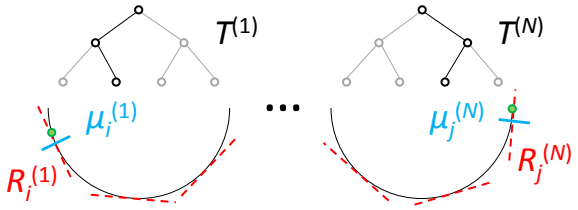

Figure 3. The main problem solved by NBSRF. Given a regression-tree ensemble, select the tree $T^{(k)}$ providing the best local linearization of the mapping function for the given datum.

4. Naive Bayes Super-Resolution Forest

Tree ensembles are well-known for many vision applications [5], but only very recently (after the submission of our original manuscript) have been employed for super resolution [20, 12]. In extension to these approaches, we propose a tree selection strategy inspired by [3]. The rationale is that combining all trees in the ensemble might not always be the best possible option. On the one hand, performing a large number of linear regressions for each datum might become prohibitively costly when the number of trees in the ensemble is large. On the other, as shown in [3] for classification tasks, combining a large number of tree decisions does not necessarily produce better results, since some trees could be deteriorating the ensemble performance (e.g. the 16 trees setup for *average* and our NBSRF in Table 3). For efficiency reasons, we aim at a scheme where just a single tree is selected from the ensemble. As we shall see in Section 5, this simple strategy, namely Naive Bayes Super-Resolution Forest (NBSRF), provides excellent performance. In Fig. 3 we illustrate the advantage of using an ensemble of trees in this manner. If we are able to quantify the selectivity of each tree $T^{(k)}, 1 \leq k \leq N$ with respect to an input datum ($\hat{x}$, green dot in the figure), we can perform a much more accurate regression than that attainable by considering a single tree. A naive solution to this problem would be to choose that tree for which the AVCS of the datum and the corresponding leaf mode is maximum. However, this criterion would discard the precious information about the space partition that leads to the leaf node in each tree and provide suboptimal performance, as reflected by the results of the *leaf*-based selection experiment in Table 3.

4.1. Data distribution

In first place, we need to model the data distribution in each node to be able to quantify the selectivity of the tree with respect to each datum. The Von Mises-Fisher distribution [7], which models distributions over a unit hypersphere, is defined as

$$f(x; \mu, \kappa) = C(\kappa) \exp\left(\kappa(\mu \cdot x)\right), \quad (7)$$

where μ is a mean direction, κ is a concentration parameter that determines the dispersion from the central mode and $C(\kappa)$ is a normalizing constant. Since our clusters are designed to contain antipodal data by exploiting the AVCS metric, the Von Mises-Fisher distribution must be correspondingly adapted:

$$f'(x;\mu,\kappa) = C'(\kappa)\exp\left(\kappa|\mu \cdot x|\right), \tag{8}$$

where $C'(\kappa)$ normalizes the modified distribution. We note that, for each node in the tree, the underlying data distribution is actually seen as a mixture of two *antipodal* Von Mises-Fisher distributions. For simplicity, in the following we shall assume that both components in the mixture have the same concentration κ, which has been empirically validated as a reasonable assumption.

4.2. Local Naive Bayes selection

In order to establish the criterion to choose the most selective tree $T^{(k*)}$ for a given patch $\hat{x}$, we rely on the Local Naive Bayes framework by McCann and Lowe [15] and conveniently adapt it to the data distribution in our tree ensemble setup. In [18] this framework was employed to select suitable image regions for adaptive dictionary learning.

Based on the original Naive Bayes derivation [4], if we assume a non-informative prior over regressors across all trees, each patch $\hat{x}$ is most accurately mapped by regressor $R_i^{(k*)}$ from tree $T^{(k*)}$ following

$$R_i^{(k*)} = \underset{R_i^{(k)}}{\operatorname{argmax}}\, p(R_i^{(k)}|\hat{x}) = \underset{R_i^{(k)}}{\operatorname{argmax}} \log p(\hat{x}|R_i^{(k)}). \tag{9}$$

If each tree has M leaf nodes (and regressors), a total number of $L = log_2(M)$ node responses or features $f_l^{(k)}$ are computed from patch $\hat{x}$ in each root-to-leaf tree traversal. The Naive Bayes assumption of feature independence results in

$$R_i^{(k*)} = \underset{R_i^{(k)}}{\operatorname{argmax}} \sum_{l=1}^{L} \log p(f_l^{(k)}|R_i^{(k)}), 1 \leq k \leq N. \tag{10}$$

The problem with this formulation is that it requires computing the likelihoods for all possible paths across the tree. Fortunately, the alternative formulation by [15] allows us to progress. The effect of each node response to a patch $\hat{x}$ can be expressed as a log-odds update. This is extremely useful for trees, since it allows us to restrict updates to only those nodes for which the descriptor gives significant evidence (i.e. the visited nodes along the root-to-leaf traversal). Let $R_i^{(k)}$ be some linear mapping and $\bar{R}_i{}^{(k)}$ the set of all other linear mappings. The odds $\mathcal{O}$ for the mapping $R_i^{(k)}$ with uniform priors is given by

$$\mathcal{O}(R_i^{(k)}) = \frac{p(R_i^{(k)}|\hat{x})}{p(\bar{R}_i{}^{(k)}|\hat{x})} = \prod_{l=1}^{L} \frac{p(f_l^{(k)}|R_i^{(k)})}{p(f_l^{(k)}|\bar{R}_i{}^{(k)})}. \tag{11}$$

The alternative classification rule expressed in terms of log-odds increments is

$$R_i^{(k*)} = \underset{R_i^{(k)}}{\operatorname{argmax}} \sum_{l=1}^{L} \log \frac{p(f_l^{(k)}|R_i^{(k)})}{p(f_l^{(k)}|\bar{R}_i{}^{(k)})}, 1 \leq k \leq N. \tag{12}$$

In the Local Naive Bayes formulation, $p(f_l^{(k)}|\bar{R}_i{}^{(k)})$ can be approximated by the likelihood of the alternative (or discarded) partition in each traversed node. The resulting method, using the antipodal Von Mises-Fisher distribution mixture and the bimodal antipodally invariant tree forest, is described in Algorithm 1. Note that, since we assume that the concentrations κ are the same for both clusters in each node, the local term in the sum of Eq. 12 reduces to $|\mu_1 \cdot \hat{x}| - |\mu_2 \cdot \hat{x}|$, where μ_1 is the most suitable mode. In the pseudocode, the node selection during the root-to-leaf traversal is implicit and $R[\mu_L^{(k*)}]$ represents the regression matrix of the chosen leaf's mode in the optimal tree.

Algorithm 1: NAIVE BAYES SR FOREST

Data: Patch $\hat{x}$, forest $\{\mu_{i,l}^{(k)}\}$, regressors $R[\mu_L^{(k)}]$
for *each tree* $k \leq N$ **do**
  $lodd^{(k)} \leftarrow 0$
  for *each level* $0 \leq l < L$ **do**
    if $|\hat{x} \cdot \mu_{1,l}^{(k)}| \geq |\hat{x} \cdot \mu_{2,l}^{(k)}|$ **then**
      $lodd^{(k)} \leftarrow lodd^{(k)} + |\mu_{1,l}^{(k)} \cdot \hat{x}| - |\mu_{2,l}^{(k)} \cdot \hat{x}|$
    else
      $lodd^{(k)} \leftarrow lodd^{(k)} + |\mu_{2,l}^{(k)} \cdot \hat{x}| - |\mu_{1,l}^{(k)} \cdot \hat{x}|$
return $k* \leftarrow \operatorname{argmax}_k lodd^{(k)}$ and $R[\mu_L^{(k*)}]$

Implementation. We would like to remark that the main performance gain of our method comes from the algorithmic choice of using a hierarchical clustering strategy, which results in a logarithimic cost in comparison to exhaustive search. Whereas the bottleneck of the approach lies on the linear regression, this is also kept to a minimum by the Naive Bayes tree selection approach (only one regression is to be computed). In order to further exploit the parallel structure of the proposed method, our MATLAB/MEX implementation makes use of parallelization. One possible venue for further improvement is to apply feature extraction and PCA compression of the input as in [31, 23, 19, 24], which shall result in smaller linear regression matrices.

5. Results

In order to assess the performance of our NBSRF, we have chosen a selection of the best performing super-resolution methods presented during the last years: the sparse coding method based on K-SVD dictionaries by

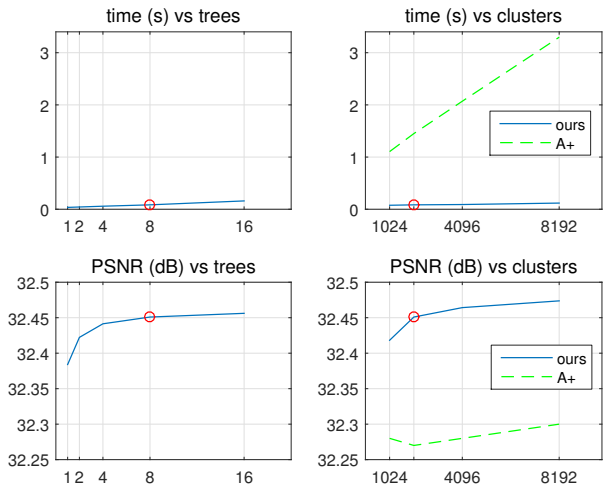

Figure 4. Average PSNR (dB) and time (s) vs. varying numbers of trees (left) and leaf nodes -or clusters- per tree (right) for 2× upscaling on Set14. The reference configuration is marked in red.

Zeyde et al. [31] (referred to as *Sparse*), the anchored neighborhood regression model by Timofte et al. [23] (*ANR*), the deep learning approach with convolutional networks proposed by Dong et al. [6] (*SRCNN*), the recently improved version of ANR [24] (*A+*) and the very recent *ASRF* with 15 trees of $L = 16$ layers [20] using the alternative training for better PSNR. All the codes have been obtained from the respective authors. For SRCNN, we have used the public implementation, which is slower than the one used in their paper, so the runtime is not meaningful. All methods use the training dataset of 91 images from [30] and, unless otherwise stated, the default configurations provided by their authors, including the 12 exponentially decaying scales of the training images in A+ and ASRF. We use a larger amount of 16 linearly decaying scales with a slope of -0.25. We have also included bicubic upscaling as a baseline.

The reference configuration of our method is a low-res patch size of 3×3 pixels, which produces progressively larger coarse inputs with the upscaling factor s (6×6 for $s = 2$, and so on). The number of trees N is set to 8 and the number of tree layers L is 11, which translates into $M = 2048$ regressors per tree. Whereas this configuration has been chosen without fine tuning, we shall see it suffices to improve over the state of the art. Despite the large scale of the parameters, the training is relatively fast, ranging from around 20 min for the $4\times$ scaling model to around one hour for the $2\times$ model on the same platform used for testing. Note that the training time would clearly be the bottleneck in applications with non-negligible unknown blur.

All methods, including our NBSRF, are run from MATLAB on an Intel Core i7 CPU equipped with 16 GB of RAM. We use the common benchmark in recent super-resolution publications [23, 6, 24, 19], consisting of $2\times$, $3\times$ and $4\times$ scaling tasks for two sets of 5 (*Set5*) and 14 (*Set14*) images of different resolutions, which we extend with the *Kodak* dataset. The latter provides 24 images free of compression artifacts with larger resolution, hence closer to a real-world scenario. In Table 1 we show the average PSNR (dB), IFC [21, 27] and runtimes obtained by each method in all scenarios. We note that our method is by far the fastest (up to one order of magnitude with respect to the runner-up A+ in $2\times$ upscaling) thanks to our fast clustering and inference strategy. More detailed results on Set5 are given in Table 2, where our method is again undisputably the fastest and the general best performing, especially for images with sharp edges like *butterfly*, where the locality of the examples used to train the regressors is more critical. In Fig. 5 we provide a qualitative comparison on the upscaling results of the best performing SR methods for some of the images in the three datasets. In general, we appreciate an enhanced robustness of our NBSRF to ringing and aliasing artifacts.

Discussion. We have deliberately kept the baseline configuration of N=8 and M=2048 used during the development of the proposed method since it suffices to showcase the excellent performance in comparison to the state of the art. However, we have also performed testing with different parameters, the most relevant of which are shown in Fig. 4.

On the left side, we show the runtime and PSNR evolution of the method with different numbers of trees. We see that the computational cost of adding more trees is linear, but with a small slope (regardless of the number of trees, only one regression takes place). The bottom left chart also serves to validate the Local Naive Bayes tree selection algorithm. The addition of more trees to the ensemble consistently produces better accuracy, even though just one of them is actually used to infer the appearance correction for the coarse patch. Whereas an even larger number of trees does not seem necessary given the saturation of the curve, we note the fair improvement obtained by the multitree approach: Switching from 1 to 8 trees provides a PSNR gain of around 0.07 dB.

On the right, we see that the addition of finer partitions in each tree does not produce a significant computational cost in NBSRF thanks to the hierarchical structure, yet it provides significant PSNR performance gains. In contrast, the overall runner-up A+ shows a prohibitive increase of the computational cost (due to the exhaustive search) and saturation symptoms when checking scalability in terms of PSNR. This result confirms the value of the bimodal tree with antipodal clustering as a powerful tool for classification and regression. It would be interesting to check the generalization to other problems beyond SR.

In Table 3 we show in more detail the effect of the contributions of the paper on $2\times$ scaling on Set14 for increasing numbers of trees with L=11 layers (M=2048 leaves).

	s	Bicubic PSNR	Bicubic IFC	Bicubic time	Sparse [31] PSNR	Sparse [31] IFC	Sparse [31] time	ANR [23] PSNR	ANR [23] IFC	ANR [23] time	SRCNN [6] PSNR	SRCNN [6] IFC	SRCNN [6] time	ASRF [20] PSNR	ASRF [20] IFC	ASRF [20] time	A+ [24] PSNR	A+ [24] IFC	A+ [24] time	NBSRF PSNR	NBSRF IFC	NBSRF time
Set5	2	33.66	5.859	0.002	35.78	7.479	1.992	35.83	7.658	0.495	36.34	7.268	3.205	36.69	8.075	0.817	36.55	7.994	0.514	**36.76**	**8.257**	**0.036**
	3	30.39	3.509	0.002	31.90	4.372	0.934	31.92	4.473	0.340	32.39	4.257	3.364	32.57	4.772	0.624	32.59	4.779	0.328	**32.75**	**4.882**	**0.053**
	4	28.42	2.302	0.001	29.69	2.889	0.559	29.69	2.951	0.227	30.09	2.822	3.106	30.20	3.131	0.705	30.28	3.186	0.225	**30.44**	**3.241**	**0.055**
Set14	2	30.23	5.898	0.002	31.81	7.329	4.015	31.80	7.445	1.101	32.18	7.053	6.515	32.36	7.780	1.427	32.28	7.736	1.105	**32.45**	**7.941**	**0.061**
	3	27.54	3.402	0.002	28.67	4.102	1.881	28.65	4.173	0.65	29.00	3.978	6.414	29.12	4.384	1.112	29.13	4.392	0.662	**29.25**	**4.484**	**0.080**
	4	26.00	2.211	0.002	26.88	2.672	1.147	26.85	2.725	0.469	27.20	2.592	6.382	27.31	2.854	0.910	27.32	2.895	0.468	**27.42**	**2.931**	**0.086**
Kodak	2	30.85	5.457	0.002	32.19	6.625	7.015	32.24	6.743	1.886	32.63	6.451	10.993	32.76	6.938	2.293	32.71	6.928	1.984	**32.81**	**7.222**	**0.105**
	3	28.43	3.116	0.002	29.22	3.679	3.300	29.21	3.740	1.139	29.43	3.658	11.065	29.53	3.859	1.728	29.57	3.889	1.154	**29.63**	**4.026**	**0.138**
	4	27.23	1.984	0.002	27.83	2.361	2.012	27.80	2.393	0.817	27.94	2.288	11.102	28.06	2.454	1.469	28.10	2.514	0.810	**28.17**	**2.590**	**0.153**

Table 1. Average PSNR (dB), IFC and runtime (s) on Set5, Set14 and Kodak for 2×, 3× and 4× scaling. Best results in bold.

Set5 images	s	Bicubic PSNR	Bicubic IFC	Bicubic time	Sparse [31] PSNR	Sparse [31] IFC	Sparse [31] time	ANR [23] PSNR	ANR [23] IFC	ANR [23] time	SRCNN [6] PSNR	SRCNN [6] IFC	SRCNN [6] time	ASRF [20] PSNR	ASRF [20] IFC	ASRF [20] time	A+ [24] PSNR	A+ [24] IFC	A+ [24] time	NBSRF PSNR	NBSRF IFC	NBSRF time
baby	2	37.1	6.15	0.002	38.2	7.43	4.678	38.4	7.64	1.114	38.3	7.23	7.536	38.5	7.74	1.633	38.5	7.73	1.126	**38.6**	**8.02**	**0.084**
bird	2	36.8	6.13	0.002	39.9	7.83	1.433	40.0	8.10	0.368	40.6	7.57	2.214	**41.2**	8.43	0.649	41.1	8.38	0.394	**41.2**	**8.61**	**0.030**
butterfly	2	27.4	5.88	0.001	30.6	8.16	1.144	30.5	8.31	0.292	32.2	8.06	1.953	32.7	9.49	0.556	32.0	9.15	0.309	**32.8**	**9.56**	**0.020**
head	2	34.9	5.47	0.001	35.6	6.65	1.354	35.7	6.74	0.350	35.6	6.37	2.161	35.7	6.84	0.619	**35.8**	6.87	0.374	**35.8**	**7.04**	**0.023**
woman	2	32.1	5.68	0.001	34.5	7.33	1.354	34.5	7.49	0.348	34.9	7.12	2.159	35.3	7.88	0.628	35.3	7.84	0.367	**35.5**	**8.05**	**0.022**
baby	3	33.9	3.70	0.004	35.1	4.40	2.200	35.1	4.50	0.801	35.0	4.16	7.589	35.2	4.58	1.232	35.2	4.59	0.762	**35.3**	**4.68**	**0.133**
bird	3	32.6	3.77	0.001	34.6	4.72	0.678	34.6	4.86	0.236	34.9	4.50	2.219	35.4	5.07	0.504	**35.5**	5.12	0.237	**35.5**	**5.17**	**0.048**
butterfly	3	24.0	3.51	0.002	25.9	4.58	0.517	25.9	4.70	0.185	27.6	4.91	1.901	27.4	5.45	0.420	27.2	5.37	0.186	**27.9**	**5.61**	**0.025**
head	3	32.9	3.15	0.001	33.6	3.76	0.631	33.6	3.85	0.264	33.5	3.50	2.280	**33.8**	3.95	0.489	**33.8**	3.98	0.235	**33.8**	**4.05**	**0.029**
woman	3	28.6	3.40	0.001	30.4	4.39	0.643	30.3	4.46	0.216	30.9	4.22	2.832	31.1	4.80	0.475	31.2	4.84	0.219	**31.3**	**4.90**	**0.029**
baby	4	31.8	2.47	0.002	33.1	2.98	1.336	33.0	3.05	0.545	33.0	2.78	7.412	33.2	3.08	1.106	33.3	3.14	0.533	**33.4**	**3.21**	**0.155**
bird	4	30.2	2.47	0.001	31.7	3.10	0.397	31.8	3.21	0.162	32.0	2.95	2.150	32.3	3.31	0.636	32.5	3.39	0.161	**32.6**	**3.42**	**0.032**
butterfly	4	22.1	2.33	0.001	23.6	3.05	0.312	23.5	3.09	0.125	**25.1**	3.39	1.842	24.6	3.55	0.570	24.4	3.52	0.124	25.0	**3.72**	**0.028**
head	4	31.6	2.01	0.001	32.2	2.42	0.371	32.3	2.47	0.151	32.2	2.21	2.049	32.4	2.52	0.600	**32.5**	**2.61**	0.153	**32.5**	**2.61**	**0.031**
woman	4	26.5	2.24	0.001	27.9	2.89	0.381	27.8	2.94	0.152	28.2	2.78	2.075	28.5	3.20	0.904	28.6	**3.27**	0.154	**28.7**	3.26	**0.031**

Table 2. PSNR (dB), IFC and runtime (s) on Set5 for 2×, 3× and 4× scaling. Best results in bold.

The first columns of SRF [20] and NBSRF essentially show the improvement of our bimodal clustering strategy. In the *bicubic* experiment we use NBSRF with bicubic interpolation instead of IBP to show the relative impact of the latter. With the *random* experiment we show that the Local Naive Bayes criterion of NBSRF is clearly better than a random tree choice (note that the latter is practically equivalent to having 1 tree), and with the *leaf* experiment we show that it is not sufficient to just observe the similarity between data and leaf modes, as pointed out in Section 4 (note that the performance is only slightly better than that achieved with a single tree). In other words, we need to exploit all the root-to-leaf computed features, as in NBSRF, to choose the optimal tree. Finally, the *average* experiment shows that carrying out all the regressions and averaging (classical random forest) provides in practice the same accuracy, yet it is costlier. The last column of this comparison shows that tree selection can even outperform averaging.

	1 tree	2 trees	4 trees	8 trees	16 trees
SRF [20]	32.11	32.16	32.21	32.21	32.22
bicubic	32.28	32.32	32.34	32.35	32.35
random	32.38	32.39	32.40	32.39	32.39
leaf	32.38	32.40	32.40	32.41	32.40
average	32.38	32.43	32.45	32.46	32.45
NBSRF	32.38	32.42	32.44	32.45	32.46

Table 3. PSNR (dB) for 2× scaling with L=11 on Set14.

6. Conclusions

In this paper we present a novel method for example-based super resolution, which we name NBSRF (Naive Bayes Super-Resolution Forest), aiming to high performance in both quality and runtime. NBSRF exploits a new interpretation of the coarse-patch space, where antipodal normalized patches (i.e. pairs of patches with the same structure but different sign) are actually part of the same clique. This observation is exploited by a hierarchical manifold learning strategy based on a partition tree with bimodal nodes, where antipodal patches are effectively clustered together and both children subnodes have comparable homogeneity, thus leading to an overall better space sampling. In order to further extend the accuracy of the local linearizations of the coarse-to-fine mapping, we further propose to use tree ensembles and select the optimal regression tree based on a Local Naive Bayes criterion. NBSRF is tested on common benchmarks and shown to improve by up to 0.9 dB in PSNR over a recent method like ANR and 0.1-0.3 dB over recent state-of-the-art methods like ASRF, A+ and SRCNN while being beyond one order of magnitude faster.

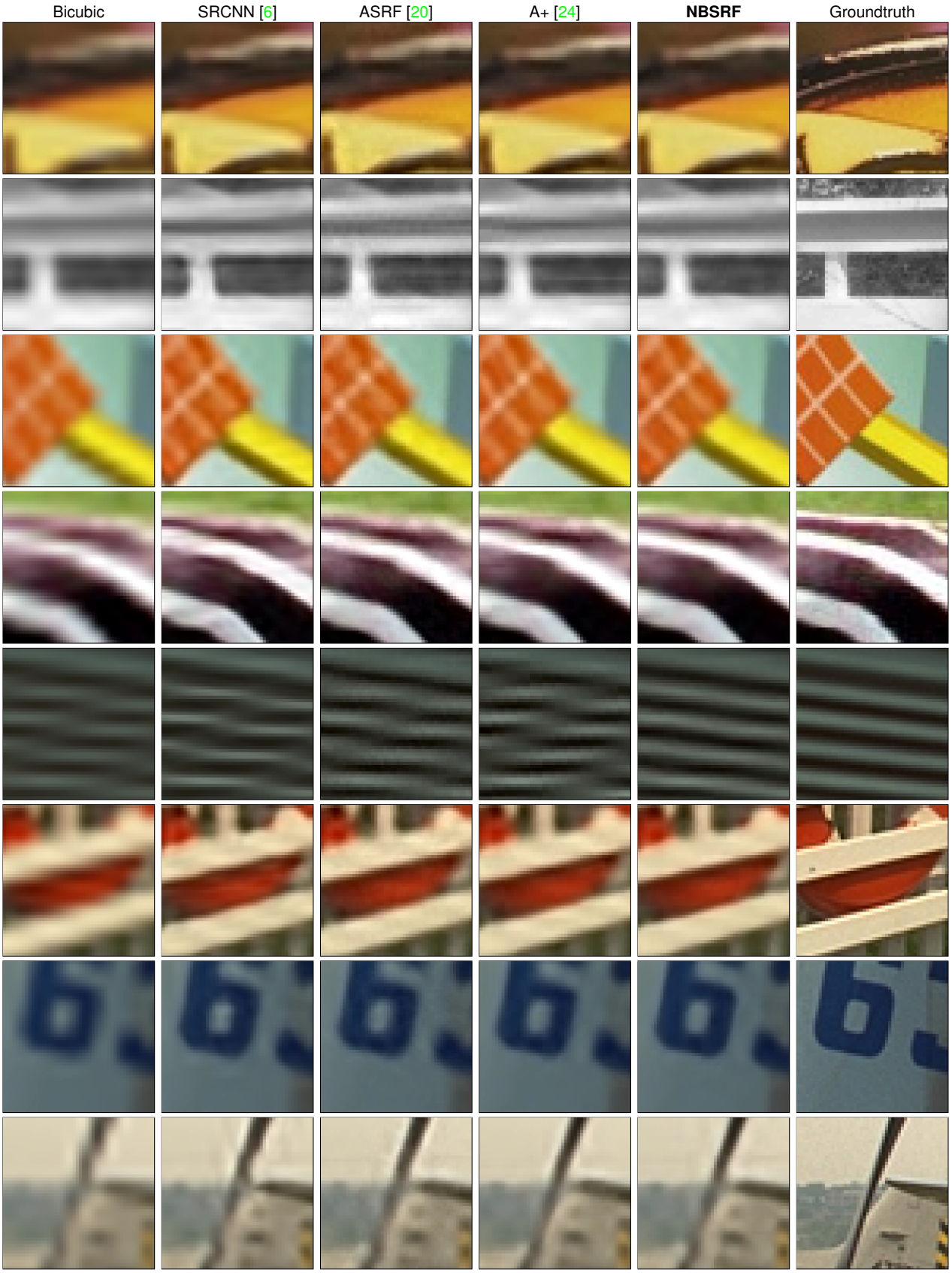

Figure 5. Close-ups of results obtained with images from Set5, Set14 and Kodak for 4× scaling. Best viewed on display.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. on Signal Processing*, 54(11):4311–4322, 2006.

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 28(3), 2009.

[3] S. Bernard, L. Heutte, and S. Adam. On the selection of decision trees in random forests. In *Proc. Int. Joint Conf. on Neural Networks*, pages 302–207, 2009.

[4] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[5] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends® in Computer Graphics and Vision*, (7):81–227, 2011.

[6] C. Dong, C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *Proc. European Conf. on Computer Vision*, pages 184–199, 2014.

[7] R. Fisher. Dispersion on a sphere. In *Proc. of the Royal Society of London; Mathematical and Physical Sciences*, 1953.

[8] G. Freedman and R. Fattal. Image and video upscaling from local self-examples. *ACM Trans. on Graphics*, 30(2):12:1–12:11, 2011.

[9] Y. Freund, S. Dasgupta, M. Kabra, and N. Verma. Learning the structure of manifolds using random projections. In *Neural Inf. Proc. Systems*, 2007.

[10] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 349–356, 2009.

[11] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S. Yoon. Spherical hashing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2012.

[12] J.-J. Huang, W.-C. Siu, and T.-R. Liu. Fast image interpolation via random forests. *IEEE Trans. on Image Processing*, 24(10):3232–3245, 2015.

[13] S. Korman and S. Avidan. Coherency sensitive hashing. In *Proc. Int. Conf. on Computer Vision*, pages 1607–1614, 2011.

[14] S. Lloyd. Least squares quantization in PCM. *IEEE Trans. on Information Theory*, 28(2):129–137, 1982.

[15] S. McCann and D. Lowe. Local Naive Bayes nearest neighbor for image classification. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 3650–3656, 2012.

[16] J. McNames. A fast nearest neighbor algorithm based on a principal axis search tree. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(9):964–976, 2001.

[17] T. Michaeli and M. Irani. Nonparametric blind super-resolution. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 945–952, 2013.

[18] E. Pérez-Pellitero, J. Salvador, J. Ruiz-Hidalgo, and B. Rosenhahn. Bayesian region selection for adaptive dictionary-based super-resolution. In *Proc. British Machine Vision Conference*, 2013.

[19] E. Pérez-Pellitero, J. Salvador, X. Torres, J. Ruiz-Hidalgo, and B. Rosenhahn. Fast super-resolution via dense local training and inverse regressor search. In *Proc. Asian Conf. on Computer Vision*, 2014.

[20] S. Schulter, C. Leistner, and H. Bischof. Fast and Accurate Image Upscaling With Super-Resolution Forests. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2015.

[21] H. Sheikh, A. Bovik, and G. de Veciana. An information fidelity criterion for image quality assessment using natural scene statistics. *IEEE Trans. on Image Processing*, 14(12):2117–2128, 2005.

[22] J. Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 111–118, 2012.

[23] R. Timofte, V. De Smet, and L. Van Gool. Anchored neighborhood regression for fast example-based super-resolution. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 1920–1927, 2013.

[24] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *Proc. Asian Conf. on Computer Vision*, 2014.

[25] I. Torres, J. Salvador, and E. Pérez-Pellitero. Fast Approximate Nearest-Neighbor Field by Cascaded Spherical Hashing. In *Proc. Asian Conf. on Computer Vision*, 2014.

[26] J. Tropp and A. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. on Information Theory*, 53(12):4655–4666, 2007.

[27] C.-Y. Yang, C. Ma, and M.-H. Yang. Single-image super-resolution: A benchmark. In *Proc. European Conf. on Computer Vision*, 2014.

[28] C.-Y. Yang and M.-H. Yang. Fast direct super-resolution by simple functions. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 561–568, 2013.

[29] J. Yang, Z. Lin, and S. Cohen. Fast image super-resolution based on in-place example regression. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1059–1066, 2013.

[30] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Trans. on Image Processing*, 19(11):2861–2873, 2010.

[31] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Proc. Int. Conf. on Curves and Surfaces*, pages 711–730, 2012.