EUROGRAPHICS 2015 / O. Sorkine-Hornung and M. Wimmer
(Guest Editors)

Volume 34 (2015), Number 2

Jointly Optimized Regressors for Image Super-resolution

D. Dai, R. Timofte, and L. Van Gool

Computer Vision Lab, ETH Zürich, Switzerland

Abstract

Learning regressors from low-resolution patches to high-resolution patches has shown promising results for image super-resolution. We observe that some regressors are better at dealing with certain cases, and others with different cases. In this paper, we jointly learn a collection of regressors, which collectively yield the smallest super-resolving error for all training data. After training, each training sample is associated with a label to indicate its 'best' regressor, the one yielding the smallest error. During testing, our method bases on the concept of 'adaptive selection' to select the most appropriate regressor for each input patch. We assume that similar patches can be super-resolved by the same regressor and use a fast, approximate kNN approach to transfer the labels of training patches to test patches. The method is conceptually simple and computationally efficient, yet very effective. Experiments on four datasets show that our method outperforms competing methods.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Image Generation—Display algorithms I.4.3 [Image Processing and Computer Vision]: Enhancement—Sharpening and deblurring

1. Introduction

Image resizing is one of the most common image operations. Almost all display and editing software employs this operation. This is necessary for example when we adapt images to displaying devices of different dimensions, when we want to explore in more details some regions of the image (e.g. in visual surveillance), when we want to map image textures to 2D/3D shapes, to name a few. The downsampling of images usually does not pose a challenge, a suitable linear pre-filtering technique doing the job. However, the upsampling of images (image super-resolution) is still an open problem. This comes from the notorious ambiguity of patch correspondence – a low-resolution (LR) image patch can be the down-sampled version of enormous high-resolution (HR) patches.

In order to reduce the ambiguity, different forms of prior knowledge have been explored. Some assume that images are smooth enough and up to some upscaling factors some interpolation formulas could provide good approximations. The assumption means that images are limited in band, and unfortunately it does not hold in most cases. Therefore, for good results, preserving only the lower frequencies from the LR image is unsatisfactory (leads to blurry, smoothed HR images and artifacts) and we need to restore or to hallucinate higher frequencies for the high-resolution image. This is possible by using more complex models and prior information from the image domain, as our proposed method does.

In this paper we focus on the exemplar-based single image super-resolution (SISR). This means that given an upscaling factor we super-resolve each input image individually by using a model which employs prior knowledge under the form of known samples of LR and corresponding HR image patches. The input LR image is decomposed over a grid in local overlapping image patches of fixed size. Each LR patch is super-resolved to an HR patch. The HR patches overlap and are averaged to create the HR output image. The prior information is extracted from natural training images as samples of LR and corresponding HR patches at the same upscaling factor. This form of prior information has been exploited by different techniques. We follow these works [NN07, KK10, TDV13, YY13] and learn it by regression functions.

As known, the regression function from LR patches to HR patches is highly non-linear. Learning a single, non-linear function to apply to every patch hardly yields satisfactory results, due to the richness of real-world image patches. There exist methods [TDV13, YY13] that address this problem by approximating this complex, non-linear function by a collection of local functions. We follow this direction and propose a novel approach where the collection of local regressors is

jointly optimized. In particular, we jointly optimize a collection of local regressors from the LR space to the HR space such that the overall super-resolving error of all patches is minimized. By doing so, it is known for sure that for any individual LR patch, there is at least one regressor which is able to yield the desired HR patch with a low error. However, for an input LR patch image the selection of the local regressor has to rely on the information from the LR image, the HR image being unknown. To tackle this, we go back to the training pool of samples for which we know the best regressors and extrapolate this information to local neighborhoods. For a neighborhood of LR patches the best regressor is the one providing the lowest cumulated super-resolving error of the LR patches. Our proposed jointly optimized regressors (JOR) together with the adaptive regressor selection for image super-resolution provide state-of-the-art quantitative performance and visual quality as shown on different real-world images in comparison with top methods.

Our paper is structured as follows. In Section 2 we briefly survey the SISR literature, focusing on those related to our approach. In Section 3 we introduce the jointly optimized regressors (JOR) technique to super-resolution. The experiments are conducted in Section 4 with discussion, which is followed by the conclusion in Section 5.

2. Previous Work

There is an impressive amount of literature addressing the single image super-resolution task. We briefly review some of the main research directions and the most relevant recent works related to our approach.

The oldest direction and very popular in commercial software is represented by the data invariant linear filters. Nearest-Neighbor, Bilinear, Bicubic, Hamming, NEDI, or Lanczos interpolation kernels are among the best known [Duc79, TBU00, LO01]. As previously said, they assume smoothness or band-limited image data, and exhibit visual artifacts such as blurring, ringing, blocking, aliasing. To address these drawbacks, one needs to use stronger prior information.

Another direction uses a strong prior information under the form of an explicit form of a distribution of energy functional over the image class. In [TRF03] the sparse derivative priors are exploited, in [TD05] the regularization PDE's, while in [DHX*07] the edge smoothness prior, and in [Fat07] the edge statistics are enforced to obtain the HR solution. Recent approaches try to estimate appropriate blur kernels instead finding good image priors [EGA*13, MI13]. Most approaches work on small image patches to then combine them in the HR output image.

In example-based super-resolution, most methods [FJP02, CYX04, YWHM10, ZEP12, TDV13, YY13] treat separately the low-frequency part and the high-frequency part of the HR image. This is motivated from the fact that the low-frequency part can be approximated reasonably well by a fast interpolation kernel such as bicubic, and the problem reduces to the estimation of the fine details, the high-frequencies, in addition to the low-frequency part. A basic way to employ priors is to extract a pool of training samples of LR and HR corresponding image patches and to infer as output the HR patch corresponding to the nearest training LR patch of the input LR patch. This is a nearest exemplar approach.

The idea of local self similarity or local image patch redundancy inside the LR image and/or across a pyramid of downscaled images is exploited in [FF11] and [GBI09]. These approaches do not need any external image patches, all the patches extracted from the LR input and used as priors already fit the image context and are meaningful to the HR image.

In [FJP02] the LR image is interpolated to the HR scale resulting a blurred image, to then add estimated high-frequency patches corresponding to the LR image and resolving their agreement through a maximum-a-posteriori (MAP) optimization. The MAP formulation provides a self-consistent solution and leads to a good performance. On the downside it relies on large training pools of patches, favorable patch candidates, and is computationally expensive.

The neighbor embedding direction starts with the method of Chang *et al.* [CYX04]. It assumes that patches naturally lies on local manifolds and the local manifolds from LR and HR spaces are in correspondence – for a local LR manifold its local HR manifold will preserve the linear relations among the samples populating it from LR to HR space. In [CYX04], for an input LR patch a local neighborhood of LR patches, its k-nearest neighbors (kNN), are extracted from the training pool. Then the input LR patch is embedded into the local coordinate system of the neighborhood using the Locally Linear Embedding (LLE) [RS01]. These coordinates are then used to reconstruct the corresponding HR output patch but this time in the corresponding HR neighborhood of patches. As shown in [TDV13], many neighbor embedding approaches which share the local manifold assumption and differ mainly in the way the input LR patch is encoded over the pool neighborhood, are able to reach comparable performance under their best settings and generally improve with the size of the training pool. In the same category, two recent approaches [TDV13, YY13] anchor the local manifolds and for each local manifold learn regressors from low to high resolution space, so that the input LR patch uses the regressor from the nearest anchored local LR manifold as offline learned. This leads to time efficiency and performance.

Another direction is represented by the sparse coding methods. They employ the sparsity assumption [YWHM08, ZEP12], that each LR patch can be sparsely decomposed over a trained dictionary of LR patches, and the HR can be

reconstructed applying the coefficients from the decomposition to the corresponding HR patches of the trained dictionary. These approaches reduce the pool of training samples to small dictionaries with large generalization capabilities, but the optimization required by the sparse decomposition can be a burden. The sparse coding methods generally produce sharp HR outputs.

A distinct category in example-based super-resolution is represented by the machine learning approaches such as Support Vector Regression (SVR) [NN07] or Convolutional Neural Networks [DLHT14]. They learn to directly regress from the LR patches to the HR patches based on a training pool of samples. Also, some SR models were proposed for specific domains such as graphics artwork [KL11].

Our proposed method resembles to the neighbor embedding methods as we work with a large pool of training samples and the local neighborhood is used to determine the regressor, while the ridge regression formulation has been already used in different frameworks for super-resolution [KK10, TDV13]. The novelty is mainly given by the joint optimization of the regressors. Our joint learning is guided by the ultimate goal of image super-resolution (minimizing the overall super-resolving error) so that the learned regressors are individually more precise and mutually complementary.

The work also shares similarity with the early work of Atkins [Atk98] and the work of Darwish and Bedair [DB96] in terms of using several regression functions based on image content. However, our main contribution is to jointly learn a collection of good regressors to choose from. Also, the selection method is different. This idea of method selection has also been used for other topics such as optical flow [LRR08] and image filtering [FKK*14].

3. Approach

Our approach consists in two main components: 1) jointly learning a fixed number of regressors, which collectively provide the smallest super-resolving error for all training images from low-resolution (LR) to high-resolution (HR); 2) adaptively selecting the most appropriate regressor for the input image patch for the super-resolution.

3.1. Jointly-optimized regressors

Motivation: There are many streams of methods to conduct image super-resolution, as summarized in Section 2. Our method follows the spirit of [KK10, TDV13] and tackles the problem as learning a regression function from LR patches to HR patches. However, because the richness of real-world image patches, as shown in [TDV13] for their global regressor, it is difficult to learn a single regressor that yields satisfactory results throughout the whole patch space. Thus, we propose to train a set of typical regressors, which collectively provide the least super-resolving error for the training set, and to select the most appropriate one for each input LR patch during test.

Data Collection: Given a set of training images, we take them as HR images, and their down-sampled versions (by a fixed factor, e.g. $\times 3$, $\times 4$) as LR images. We then decompose all LR images to small patches $\mathbf{x}$ and look for their corresponding HR patches $\mathbf{y}$ to collect our training examples: $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_L, \mathbf{y}_L)\} \in \mathbb{R}^M \times \mathbb{R}^N$. The representation of $\mathbf{x}$ and $\mathbf{y}$ will be elaborated in Section 4.3.

Training: Once having the training examples, our goal is to discover a fixed number of O regressors $\mathcal{F} = \{f_1, ..., f_O\}$ that collectively give the least super-resolving error for all the patches, each patch being reconstructed by its most appropriate regressor. The challenge is that the space of the regression functions is enormous, and we must sift through all the data to find the ones which are individually representative and mutually complementary. We pose this problem as a clustering problem: given millions of patch pairs, we group them so that patches in the same cluster can be reconstructed precisely by the same regressor (one of the O regressors). Note that we do not impose the constraint that patches in the same cluster must be similar in appearance; patches can share the same regression function even they reside far apart from each other in feature space.

Mathematically, the problem is formulated as follows. Given the collection of pairs of patches $\mathcal{D}$, our goal is to cluster the data into O clusters and to learn O regressors, one for each cluster. Cluster o is associated with a regressor f_o and an indicator vector $\mathbf{c}_o \in \{0,1\}^L$ with $c_{o,l} = 1$ for instance l being in cluster o and zero otherwise. Hard clustering is performed: each image patch only belongs to one cluster. Ideally, we would like to minimize the following function:

$$\min_{\mathbf{C},\mathcal{F}} \sum_{l=1}^{L} \sum_{o=1}^{O} c_{o,l} \|f_o(\mathbf{x}_l) - \mathbf{y}_l\|^2 \tag{1}$$

where we cumulate the super-resolving error, $\mathbf{C}$ denotes the matrix form of all O membership indicator vectors, i.e. $\mathbf{C} = [\mathbf{c}_1, ..., \mathbf{c}_O]$.

Exactly minimizing the objective function in Eq.1 is very challenging, as this is a chicken-egg problem: if we know the optimal membership indicator $\mathbf{c}_o$, $o \in \{1, ..., O\}$, we can compute the optimal regressor f_o with all samples in cluster o, and if we had the optimal regressors f_o, $\mathbf{c}_o$ can be determined by setting 1 to the patches to which f_o gives the smallest super-resolving error among all regressors. In this paper, we follow the spirit of EM algorithm and propose to solve this problem iteratively. Given $\mathbf{c}_o, o \in \{1, ..., O\}$, we compute the regressor f_o using the patch pairs within cluster o (an analogue to the E-step of EM algorithm). Once we have regressors $\{f_o\}$, we can refine the membership indicators $\{\mathbf{c}_o\}$ (an analogue to the M-step of EM algorithm). We repeat the E-M steps until convergence. Note that E-step is used here with a slight abuse of terminology as there is no

Algorithm 1: Jointly Optimized Regressors

Data: $\mathcal{D}$, O, and λ
Result: $\mathcal{F}$, $\mathbf{C}$ and $\mathbf{Z}$
begin
  Initialize $\mathbf{C}$ by running K-means to cluster all LR patches $\{\mathbf{x}_l\}$ to O clusters.
  while *not converged (with tolerance error)* **do**
    E-step: Estimate the O regressors, $\{f_o\}$, for each cluster indicated by $\mathbf{C}$ using Eq.(2);
    M-step: Refine the clusters ($\mathbf{C}$) according to the regressor f_o which provides the minimum error (4) for each sample $(\mathbf{x}_l, \mathbf{y}_l)$;
  end
end

probabilistic expectation to maximize in our E-step. The initial $\mathbf{c}_o$ are obtained by running K-means clustering method on all the LR training patches $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_L]$.

For the regressors, we follow [KK10, TDV13] and use Ridge Regression (RR). The main benefit of using Ridge Regression is that it is simple and has a closed-form solution. By doing this, the learning process can scale easily to millions of samples, which is challenging for non-linear techniques such as Supported Vector Regression. However, our framework is agnostic to the type of regressors. The solution of RR is computed offline and saved as a simple projection matrix to apply to new image patches. The projection matrix for f_o can be written as:

$$\mathbf{P}_o = \mathbf{Y}_o(\mathbf{X}_o^T\mathbf{X}_o + \lambda\mathbf{I})^{-1}\mathbf{X}_o^T \tag{2}$$

where $\mathbf{X}_o$ are the LR patches from the o-th cluster stacked column-wise, $\mathbf{Y}_o$ are the corresponding HR patches, and λ is a regulatory parameter fixed to 0.1 in our experiments. The identity matrix is added mainly to regularize the solution, i.e. to avoid overfitting, which is derived from the regularization term of Ridge Regression. The term is also useful in handling the presence of non-singular matrices, when highly redundant samples (e.g. smooth ones) occur in the training set. By doing so, the super-resolving of a LR patch $\mathbf{x}_l$ by a regressor f_o is reduced to a simple matrix projection:

$$\tilde{\mathbf{y}}_{o,l} = f_o(\mathbf{x}_l) = \mathbf{P}_o\mathbf{x}_l \tag{3}$$

and its super-resolving error is signaled as follows:

$$z_{o,l} = \| \tilde{\mathbf{y}}_{o,l} - \mathbf{y}_l \|^2 \tag{4}$$

The super-resolving error of all regressors f to all patches $\mathbf{x}$ is recorded in $\mathbf{Z} = [\mathbf{z}_l, ..., \mathbf{z}_L]$, where $\mathbf{z}_l \in \mathbb{R}^O$ recording the super-resolving error of $\mathbf{x}_l$ by all the O regressors. The learning algorithm is detailed in Algorithm 1. We interchangeably use $\mathbf{P}_o$ and f_o for the oth regressor.

3.2. Upsampling with JOR

After training, high resolution (HR) patches $\mathbf{y}$ in $\mathcal{D}$ are disgarded. Each low resolution (LR) patch $\mathbf{x}_l$ of the training data is associated with a vector $\mathbf{z}_l$. During test, our method decomposes LR images to small patches over a grid (same size as used in training) and predicts the most appropriate regressors for each single patch. With labels to the regressors, LR patches are reconstructed to HR ones by performing the matrix projection as shown in Eq.3. Reconstructed HR patches are assembled to go back to the desired HR image. Below, we elaborate the method for adaptive selection of regressors.

The selection is performed in two steps for a given LR patch: 1) the super-resolving error of all regressors for the patch is estimated; 2) the regressor with the smallest error is then chosen for super-resolving. In order to handle the intricate dependencies between patch content and regressor annotations, we employ non-parametric methods. The k-nearest neighbor method is used in this work. The selection is based on local super-resolving accuracies in a 'neighborhood' of the input patch $\mathbf{x}$. The neighborhood $\mathcal{N}(\mathbf{x}) = \{n_1, ..., n_K\}$ is defined with respect to the training sets $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_L\}$, where $n_k \in \{1, ..., L\}$ is the index of a training patch and K denotes the size of neighborhood. Specifically, the super-resolving error for patch $\mathbf{x}$ by regressor f_o is estimated as

$$z_o = \frac{\sum_{k=1}^{K} \frac{1}{k} z_{o,n_k}}{\sum_{k=1}^{K} \frac{1}{k}}, \tag{5}$$

where $\frac{1}{k}$ is used to modulate the contribution of neighbors, and it improves performance slightly over uniform weights. The regressor with smallest error is used to super-resolve the patch $\mathbf{x}$. Our adaptive selection is based on the super-resolving error accumulated over a small neighborhood, which can be taken as a 'soft' version of the normal kNN method. It is especially useful for samples which are close to borders of the clusters. We use k-d tree (the implementation in the VLFeat library [VF08]) to organize the traing samples for fast, approximate kNN search.

4. Experiments

4.1. Datasets

For training, we follow existing work [YWHM08, ZEP12, TDV13, DLHT14] and use the standard training set, which contains 91 images. For test, we use four datasets, including three from [TDV13] and a newly compiled one. The three datasets are Set5, Set14 and BD100, containing 5, 14, and 100 images respectively. Set14 was proposed by Zeyde *et al.* [ZEP12], while BD100 dataset contains the 100 testing images from the Berkeley Segmentation Dataset (BSDS300) [MFTM01]. In order to further evaluate the ability of all methods for texture recovery, a highly-desired property for image super-resolution, we created a new dataset

by selecting 136 diverse texture images from the ETH-Synthesizability dataset [DRV14]. The dataset is named SuperTex136. The textures were chosen to cover a large range of materials (e.g. metal, plastic, glass, water) and geometrical properties (e.g. stochastic, lined, structured) of textures.

4.2. Compared methods

In our comparison we consider all the methods considered in [TDV13], and in addition we compare with the Convolutional Neural Network method (SRCNN) proposed by Dong *et al.* [DLHT14]. The methods in [TDV13] are: Neighbor Embedding with Least Squares decomposition (NE+LS), Neighbor Embedding with Locally Linear Embedding (NE+LLE) [CYX04], Neighbor Embedding with Non-Negative Least Squares decomposition (NE+NNLS), Zeyde *et al.* [ZEP12], and the Global Regression (GR) and Anchored Neighborhood Regression (ANR) methods of Timofte *et al.* [TDV13]. All these methods share the same training material and share the same training dictionary with an exception to SRCNN where no dictionary is used. Our JOR method uses the same training dataset as these methods. The methods are compared quantitatively in terms of Peak Signal-to-Noise Ratio (PSNR) of their HR output images. PSNR usually correlates well with the visual quality.

4.3. Features

We follow the data representation and the general framework from [ZEP12, TDV13]. The basic idea is that the low-frequency part can be approximated reasonably well by a fast interpolation kernel such as bicubic, thus the problem reduces to the estimation of the fine details. It has also been found that gradient features are most relevant to high-resolution details. Therefore, the LR and HR patches are represented as follows. In the training, the HR images are downscaled to the LR corresponding images for a given upscaling factor. Then, the LR image is bicubically interpolated by the same factor to get to an interpolated HR image. The first and second order gradient filters are applied vertically and horizontally to this image, and the LR image patches $\mathbf{x}$ are represented as the concatenation of corresponding gradient responses. Note that the interpolated HR images are also called LR images, as the high-frequency details are still missing.

The HR patches $\mathbf{y}$ are represented by the difference between the true HR image patches and the interpolated ones from their corresponding LR patches. The LR patches are of 3×3 pixels, i.e. for upscaling factors of $\times 2$, $\times 3$, and $\times 4$ we work with patches of 6×6, 9×9, and 12×12 pixels, respectively. Since, the representation of LR patches is quite high-dimensional, we apply the PCA projection to reduce the dimensionality by preserving 99.9% of the average energy. The PCA is learned over the training patches, and the same PCA projection is applied during testing. This typically leads to 30 dimensions for an initial 324-dimensional representation, for upscaling factor $\times 3$. The 3×3 patches are

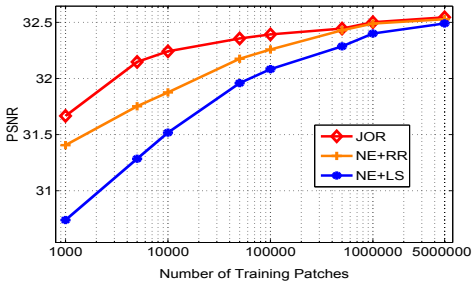

Figure 1: *PSNR as a function of the number of training patches, on Set5 with factor* $\times 3$.

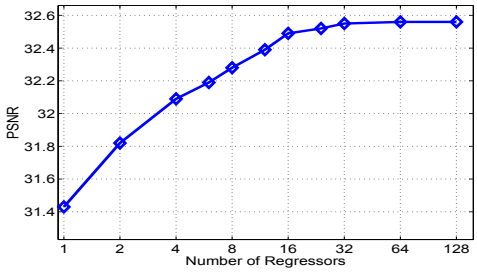

Figure 2: *PSNR as a function of the number of regressors, on Set5 with factor* $\times 3$.

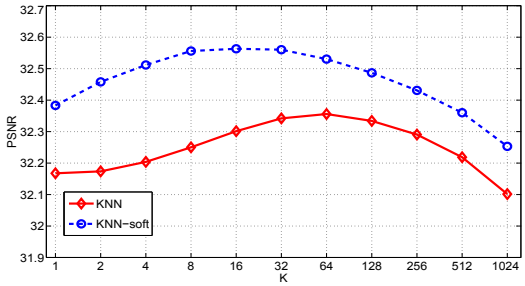

Figure 3: *PSNR as a function of K on Set5 with factor* $\times 3$, *where our soft kNN is compared with the normal kNN in regressor selection.*

extracted over a regular grid with an overlap of 2 pixels. For training samples, we first extract from the original training images and then repeat the extraction procedure after bicubically downscaling the images with a factor of 0.98 until we achieve the desired number of training patches. Note the down-sampling step is not necessary if a large number of training images are used. During testing, the LR input image is first bicubically interpolated, and then over a dense grid the LR patch features are computed. Then the LR features are super-resolved to the HR outputs and added to the interpolated content to form the HR output image. All the methods we compare with share the same features and pipeline, except for SRCNN [DLHT14].

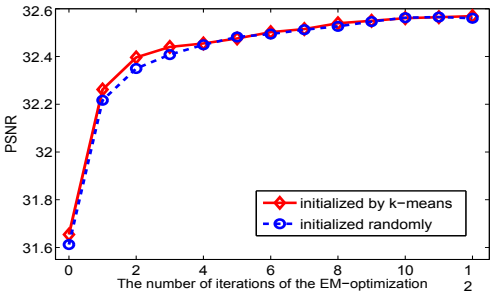

Figure 4: *PSNR as a function of the number of iterations of the EM-optimization, on Set5 with factor ×3. Iteration 0 denotes its initialization, which is performed by k-means with a comparison here to a random initialization.*

4.4. Parameters

Our JOR method comes with a number of parameters. Here, we investigate the influence of four main parameters on the performance. They are the number of training patches L, the number of regressors O, K of the kNN method, and the number of iterations of our EM-optimization. The results are shown in Fig. 1, Fig. 2, Fig. 3, and Fig. 4, respectively.

In Fig. 1 we vary the amount of training samples extracted from the training images from as few as 1000 samples to 5 million samples and report the PSNR average performance on Set5 dataset. As expected, the larger the number of samples are, the better is the coverage of the LR space and the more accurate is the prediction of the HR output. JOR improves from 31.7dB when 1000 samples are used to over 32.55 for 5 million samples. In comparison we report the performance of two neighbor embedding methods (NE+LS and NE+RR) with the same set of training patches. The NE+LS as defined in [TDV13] uses the least squares to encode the input LR patch, and then to reconstruct the HR output patch from the corresponding HR patches of the neighborhood. The NE+RR is a variant of NE+LS, using instead a ridge regression decomposition as in ANR [TDV13] and in our method. The figure shows that NE+LS and NE+RR benefit as well from the increased number of samples, and, given sufficient training samples (e.g. 5 million samples) will eventually converge to a performance comparable to our JOR method. However, both NE+LS and NE+RR are more computational demanding. In comparison with JOR, NE+LS needs to retrieve a neighborhood and solve the least squares online for every LR input patch, while NE+RR needs to solve a RR. This renders them much slower than JOR when the same number of training samples are used. Furthermore, JOR is more efficient by handling orders of magnitude fewer samples for comparable performance to NE+LS or NE+RR.

The number of joint regressors is another parameter of JOR, the more regressors the smaller the theoretical achievable errors between the estimated HR patches and the ground truth ones. As shown in Fig. 2, JOR is quite robust with respect to the number of regressors. JOR's performance faces a plateau at 32 joint regressors, above this, a larger number of regressors does not improve significantly. This early convergence is due to the fact that the adaptive selection of regressors is more challenging when the number of regressors increases. However, JOR working with 32 regressors significantly outperforms the most similar method ANR [TDV13] and yields comparable results with A+ [TDV14], both of which work with 1024 regressors (See details in Sec.4.5). This suggests that the regressors learned by our joint learning are more powerful than the regressors trained in previous works. In order to use more regressors, a more sophisticated classification system is needed. For instance, one can learn a more accurate distance metric by the Metric Learning technique instead of using the Euclidean distance directly.

The size of neighborhood K also influences the performance, but we found that JOR is quite robust to it. See Fig. 3 for the results. We tested a wide range of its values from 1 to 1024 and found $K = 16$ yields the best performance, but the difference from different values of K falls within 0.3. We also compared our soft kNN with the normal kNN, where the labels are directly used for regressor selection. The figure shows that our soft strategy by accumulating error over a small neighborhood consistently improves the performance of kNN for all values of K, e.g. by 0.25 dB when $K = 16$. It is mainly because a large amount of information is lost when the reconstruction errors of regressors are converted to crisp labels of regressors. It has also been observed in other vision tasks, e.g. in the feature learning of bag-of-words, that the soft-version of kNN beats its hard-version.

As to the influence of the EM-optimization, Fig. 4 shows that the performance improves with the number of the iteration of the EM. The performance grows very quickly at the beginning and starts to plateau after 10 iterations. The total improvement is around 1 dB. We also tested the influence of our initialization by k-means by a comparison to a random initialization. It is found that k-means performs only slightly better than the random one, and the benefit has vanished after several rounds of iterations. This suggests that our EM method is robust to the initialization, and is able to generate good results even with a totally random initialization. It also suggests that a simple clustering (e.g. k-means) based on general features cannot partition well the space of the regression functions. Our optimization (clustering), which is guided by the ultimate goal of image super-resolution – minimizing the overall super-resolving error of all training patches – performs significantly better than conventional clustering methods.

From these observations, in all our following experiments, JOR is by default set to use 5 million training samples, 32 joint regressors learned by our EM method with 10 iterations, and a soft kNN method with neighborhood size of 16.

Table 1: *Average PSNR on Set5, Set14, BD100, and SuperTex136.*

Benchmark		Bicubic	Zeyde *et al.* [ZEP12]	GR [TDV13]	ANR [TDV13]	NE+LS [TDV13]	NE+NNLS [TDV13]	NE+LLE [TDV13]	SRCNN [DLHT14]	**JOR (Ours)**
Set5	x3	30.39	31.90	31.41	31.92	31.78	31.60	31.84	32.39	**32.55**
	x4	28.42	29.69	29.34	29.69	29.55	29.47	29.61	30.09	**30.19**
Set14	x3	27.54	28.67	28.31	28.65	28.59	28.44	28.60	29.00	**29.09**
	x4	26.00	26.88	26.60	26.85	26.81	26.72	26.81	27.20	**27.26**
BD100	x3	27.15	27.87	27.70	27.89	27.83	27.73	27.85	28.10	**28.17**
	x4	25.92	26.51	26.37	26.51	26.45	26.41	26.47	26.66	**26.74**
SuperTex136	x3	24.63	25.33	24.97	25.38	25.34	25.31	25.36	25.47	**25.59**
	x4	23.90	24.52	24.22	24.54	24.51	24.48	24.45	24.61	**24.73**

4.5. Performance

To asses the performance of our JOR method, we use upscaling factors of $\times 3$ and $\times 4$, and report PSNR results on the four datasets. The average results of JOR and other state-of-the-art methods are reported in Table 1. Note that due to the computational demands we decided not to report results of NE+RR. However, the previous experiment shown that NE+RR and NE+LS in our settings can reach comparable performance with JOR but at much higher computational cost. The table shows that JOR consistently improves over all of the compared methods. The improvements over them are also very considerable, except for SRCNN. However, for factor x3, the gap between JOR and SRCNN are still 0.16dB on Set5, 0.09dB on Set14, 0.07 on BD100, and 0.12 on SuperTex136. For factor x4, they are 0.10dB, 0.06dB, 0.08dB, and 0.12dB, respectively. The differences show clear improvement. Also, we should be aware that making further improvement is very hard when the performance has already been pushed very high by previous methods, e.g. by SRCNN method here. The table also shows that JOR performs especially well for textures. This is mainly because JOR partitions training patches into groups of specific patches and trains specialized regressors. The strategy is more helpful for complex visual patterns such as textures where a universal transformation function hardly works well for all samples. Performance on textures is important as enriching the missing textures of LR images is one of the ultimate goal of image super-resolution.

JOR is efficient at testing, as the regression functions are computed offline and the system only searches k-nearest neighbors (kNN) during test time. This significantly speedups the system, as searching kNN can be done efficiently with the help of organized data structures. We use k-d tree in our experiment. For the case of 5 million patches, the speed of our system is around $2 \sim 3$ seconds for an image of 400×400 pixels for upscaling factor $\times 3$, which is on par with the speed of SRCNN method.

As to training time, we see a clear advantage for our JOR method over the SRCNN. JOR takes tens of minutes to train the regressors on a desktop PC without using any GPU, while SRCNN takes 3 days with a GPU. Also, our method is conceptually simpler than SRCNN. The framework is flexible for adaptation, i.e. it is easy to replace some parts by more sophisticated alternatives, such as the linear regressors by non-linear ones, the k-d tree by hashing functions.

4.6. Visual Quality

As most of the methods of image super-resolution, JOR is introduced using a single luminance channel. The RGB images were converted to YCbCr color space, and the HR output color image is reconstructed by bicubically interpolating the chrominance channels (Cb and Cr) and super-resolving the luminance information (Y) of the input LR image.

In Fig. 5, Fig. 6 and Fig. 7, we show results with an upscaling factor $\times 3$ for natural images and textures from the four datasets. In Fig. 8 we show the results with an upscaling factor x4. In most cases, JOR is compared to ANR and SRCNN, as they perform the best among all existing methods. The figures show that JOR generally yields better visual results, which is consistent with the PSNR results. Taking the 'Foreman' image in Fig. 5 and the '210088' in Fig. 8 as examples, JOR generates sharper edges with fewer ringing artifacts. As the PSNR shows, JOR also works well for textures. This can be verified by the results in Fig. 7 and Fig. 8. For instance, JOR is better in recovering the details of the woven fabric and the beans, the last two images in Fig. 8.

5. Conclusions

We have introduced a simple and efficient method for image super-resolution. Under the guidance of the ultimate goal of image super-resolution, we have proposed to jointly learn a collection of local functions (regressors) which minimize the overall super-resolving error of all training data, and to re-solve each test patch by its most appropriate regressor. Results on four datasets show that the method outperforms competing methods. The code, the dataset SuperTex136, and more results are available at `www.vision.ee.ethz.ch/~daid/JOR`.

Acknowledgement. The work is supported by the ERC Advanced Grant VarCity and the ETH-Singapore project Future Cities.

Original / PSNR | Bicubic / 32.6 | SRCNN / 34.9 | JOR / **35.5**

Original / PSNR | Bicubic / 31.2 | SRCNN / 33.3 | JOR / **34.0**

Original / PSNR | Bicubic / 32.4 | SRCNN / 34.4 | JOR / **34.7**

Figure 5: *Top to Bottom: the 'Bird' from Set5, and the 'Foreman' and the 'Pepper' from Set14 with an upscaling factor* $\times 3$.

Original / PSNR | Bicubic / 27.9 | Zeyde et al. / 28.7

ANR / 28.8 | SRCNN / 29.0 | JOR / **29.3**

Figure 6: *The results of image 37073 from BD100 with an upscaling factor* $\times 3$.

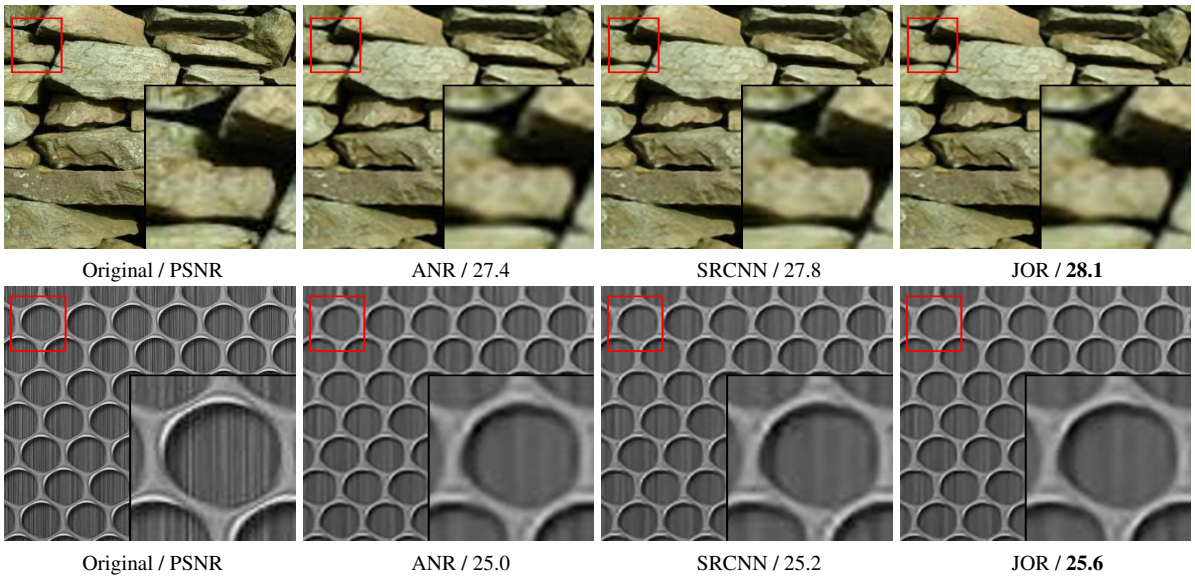

Figure 7: *Top to Bottom: Stone_235 and Metal_68 from SuperTex136. with an upscaling factor* $\times 3$.

References

[Atk98] ATKINS C. B.: *Classification-based methods in optimal image interpolation*. PhD thesis, Purdue University, 1998. 3

[CYX04] CHANG H., YEUNG D.-Y., XIONG Y.: Super-resolution through neighbor embedding. *CVPR* (2004). 2, 5

[DB96] DARWISH A. M., BEDAIR M. S.: Adaptive resampling algorithm for image zooming. In *Electronic Imaging: Science & Technology* (1996), pp. 131–144. 3

[DHX*07] DAI S., HAN M., XU W., WU Y., GONG Y.: Soft edge smoothness prior for alpha channel super resolution. In *CVPR* (2007). 2

[DLHT14] DONG C., LOY C. C., HE K., TANG X.: Learning a deep convolutional network for image super-resolution. In *ECCV* (2014). 3, 4, 5, 7

[DRV14] DAI D., RIEMENSCHNEIDER H., VAN GOOL L.: The synthesizability of texture examples. In *CVPR* (2014). 5

[Duc79] DUCHON C. E.: Lanczos Filtering in One and Two Dimensions. *J. Appl. Meteorology 18* (1979), 1016–1022. 2

[EGA*13] EFRAT N., GLASNER D., APARTSIN A., NADLER B., LEVIN A.: Accurate blur models vs. image priors in single image super-resolution. In *ICCV* (2013). 2

[Fat07] FATTAL R.: Image upsampling via imposed edge statistics. *ACM Trans. Graph. 26*, 3 (2007). 2

[FF11] FREEDMAN G., FATTAL R.: Image and video upscaling from local self-examples. *ACM Trans. Graph. 30*, 2 (2011), 12:1–12:11. 2

[FJP02] FREEMAN W. T., JONES T. R., PASZTOR E. C.: Example-based super-resolution. *IEEE Computer Graphics and Applications 22*, 2 (2002), 56–65. 2

[FKK*14] FANELLO S., KESKIN C., KOHLI P., IZADI S., CN J. S., CRIMINISI A., PATTACCINI U., PAEK T.: Filter forests for learning data-dependent convolutional kernels. In *CVPR* (2014). 3

[GBI09] GLASNER D., BAGON S., IRANI M.: Super-resolution from a single image. In *ICCV* (2009). 2

[KK10] KIM K. I., KWON Y.: Single-image super-resolution using sparse regression and natural image prior. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 6 (2010), 1127–1133. 1, 3, 4

[KL11] KOPF J., LISCHINSKI D.: Depixelizing pixel art. In *ACM SIGGRAPH* (2011). 3

[LO01] LI X., ORCHARD M. T.: New edge-directed interpolation. *IEEE Trans. Image Process. 10*, 10 (2001), 1521–1527. 2

[LRR08] LEMPITSKY V., ROTH S., ROTHER C.: Fusionflow: Discrete-continuous optimization for optical flow estimation. In *CVPR* (2008). 3

[MFTM01] 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. In *ICCV* (2001). 4

[MI13] MICHAELI T., IRANI M.: Nonparametric blind super-resolution. In *ICCV* (2013). 2

[NN07] NI K., NGUYEN T.: Image superresolution using support vector regression. *IEEE Trans. Image Process. 16*, 6 (2007), 1596–1610. 1, 3

[RS01] ROWEIS S., SAUL L.: Nonlinear dimensionality reduction by locally linear embedding. In *ICCV* (2001). 2

[TBU00] THÉVENAZ P., BLU T., UNSER M.: *Image interpolation and resampling*. 2000. 2

[TD05] TSCHUMPERLE D., DERICHE R.: Vector-valued image regularization with pdes: a common framework for different applications. *IEEE Trans. Pattern Anal. Mach. Intell. 27*, 4 (2005), 506–517. 2

[TDV13] TIMOFTE R., DE SMET V., VAN GOOL L.: Anchored neighborhood regression for fast example-based super resolution. In *ICCV* (2013). 1, 2, 3, 4, 5, 6, 7

[TDV14] TIMOFTE R., DE SMET V., VAN GOOL L.: A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV* (2014). 6

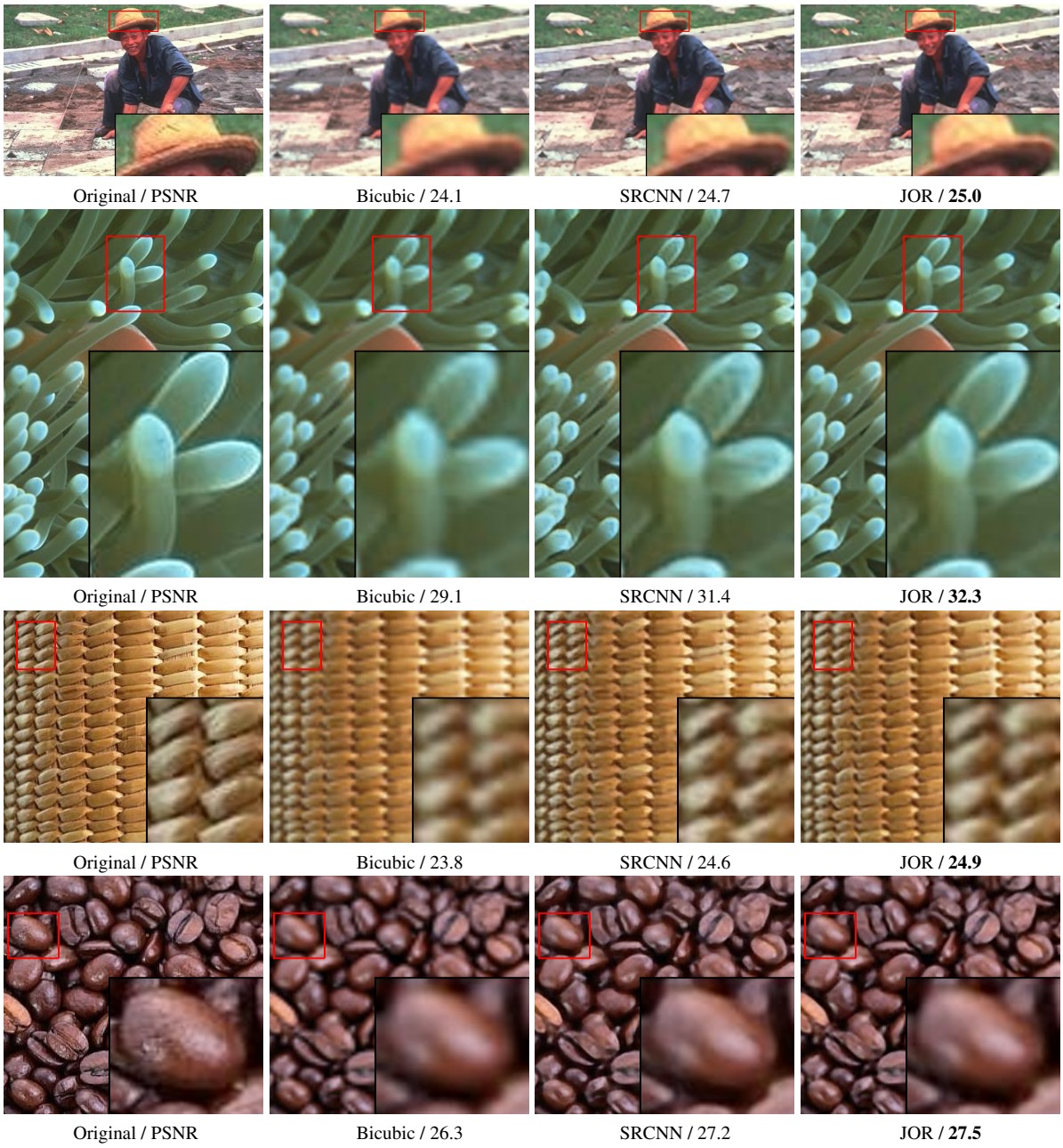

Figure 8: *Top to bottom: '85048' and '210088' from BD100, Woven_191 and Texture_667 from SuperTex136., with factor* $\times 4$.

[TRF03] TAPPEN M. F., RUSSELL B. C., FREEMAN W. T.: Exploiting the sparse derivative prior for super-resolution and image demosaicing. In *In IEEE Workshop on Statistical and Computational Theories of Vision* (2003). 2

[VF08] VEDALDI A., FULKERSON B.: VLFeat: An open and portable library of computer vision algorithms, 2008. 4

[YWHM08] YANG J., WRIGHT J., HUANG T. S., MA Y.: Image super-resolution as sparse representation of raw image patches. In *CVPR* (2008). 2, 4

[YWHM10] YANG J., WRIGHT J., HUANG T., MA Y.: Image super-resolution via sparse representation. *IEEE Trans. Image Process. 19*, 11 (2010), 2861–2873. 2

[YY13] YANG C.-Y., YANG M.-H.: Fast direct super-resolution by simple functions. In *ICCV* (2013). 1, 2

[ZEP12] ZEYDE R., ELAD M., PROTTER M.: On single image scale-up using sparse-representations. In *Curves and Surfaces* (2012), pp. 711–730. 2, 4, 5, 7