Fast Super-Resolution via Dense Local Training and Inverse Regressor Search

Eduardo Pérez-Pellitero[1,2], Jordi Salvador[1], Iban Torres-Xirau[1], Javier Ruiz-Hidalgo[3], and Bodo Rosenhahn[2]

[1] Technicolor R&I Hannover
[2] TNT Lab, Leibniz Universität Hannover
[3] Image Processing Group, Universitat Politècnica de Catalunya

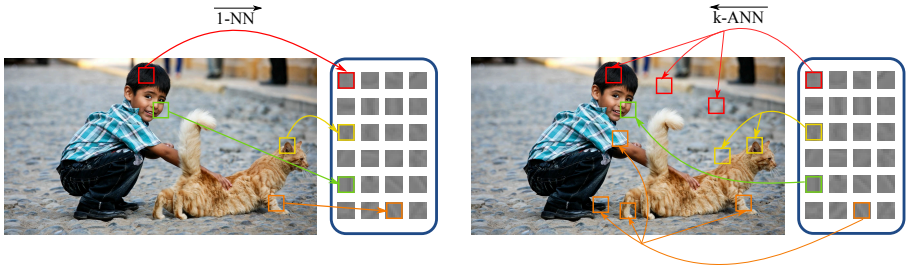
Abstract. Regression-based Super-Resolution (SR) addresses the upscaling problem by learning a mapping function (i.e. regressor) from the low-resolution to the high-resolution manifold. Under the locally linear assumption, this complex non-linear mapping can be properly modeled by a set of linear regressors distributed across the manifold. In such methods, most of the testing time is spent searching for the right regressor within this trained set. In this paper we propose a novel inverse-search approach for regression-based SR. Instead of performing a search from the image to the dictionary of regressors, the search is done inversely from the regressors' dictionary to the image patches. We approximate this framework by applying spherical hashing to both image and regressors, which reduces the inverse search into computing a trained function. Additionally, we propose an improved training scheme for SR linear regressors which improves perceived and objective quality. By merging both contributions we improve speed and quality compared to the state-of-the-art.

1 Introduction

Super resolution (SR) comprises any reconstruction technique capable of extending the resolution of a discrete signal beyond the limits of the corresponding capture device. The SR problem is by nature ill-posed, so the definition of suitable priors is critical. Over more than two decades, many of them have been proposed.

Originally, image SR methods were based on piecewise linear and smooth priors (i.e. bilinear and bicubic interpolation, respectively), resulting in fast interpolation-based algorithms. Tsai and Huang [1] showed that it was possible to reconstruct higher-resolution images by registering and fusing multiple images, thus pioneering a vast amount of approaches on multi-image SR, often called reconstruction-based SR. This idea was further refined, among others, with the introduction of iterative back-projection for improved registration by Irani and Peleg [2], although further analysis by Baker and Kanade [3] and Lin and Shum [4] showed fundamental limits on this type of SR, mainly conditioned by registration accuracy. Learning-based SR, also known as example-based, overcame

some of the aforementioned limitations by avoiding the necessity of a registration process and by building the priors from image statistics. The original work by Freeman et al. [5] aims to *learn* from patch- or feature-based examples to produce effective magnification well beyond the practical limits of multi-image SR.



(a) Previous approaches (b) Proposed approach

Fig. 1: Overview of the proposed inverse-search SR: (a) Previous approaches search the 1st nearest dictionary atom for each image patch. (b) Our Proposed approach searches the k-nearest image patches for each dictionary atom.

Example-based SR approaches using dictionaries are usually divided into two categories: internal and external dictionary-based SR. The first exploits the strong self-similarity prior. This prior is learnt directly from the relationship of image patches across different scales of the input image. The opening work on this subcategory was introduced by Glasner et al. [6], presenting a powerful framework for fusing reconstruction-based and example-based SR. Further research on this category by Freedman and Fattal [7] introduced a mechanism for high-frequency transfer based on examples from a small area around each patch, thus better localizing the cross-scale self-similarity prior to the spatial neighborhood. The recent work of Yang et al. [8] further develops the idea of localizing the cross-scale self-similarity prior arriving to the in-place prior, i.e. the best match across scales is located exactly in the same position if the scale is similar enough.

External dictionary-based SR methods use other images to build their dictionaries. A representative widely used approach is the one based on sparse decomposition. The main idea behind this approach is the decomposition of each patch in the input image into a combination of a sparse subset of entries in a compact dictionary. The work of Yang et al. [9] uses an external database composed of related low and high-resolution patches to jointly learn a compact dictionary pair. During testing, each image patch is decomposed into a sparse linear combination of the entries in the low-resolution (LR) dictionary and the same weights are used to generate the high-resolution (HR) patch as a linear combination of the HR entries. Both the dictionary training and testing are

costly due to the L_1 regularization term enforcing sparsity. The work of Zeyde et al. [10] extends sparse SR by proposing several algorithmic speed-ups which also improves performance. However, the bottleneck of sparsity methods still remains in the sparse decomposition.

More recently, regression-based SR has received a great deal of attention by the research community. In this case, the goal is to learn a certain mapping from the manifold of the LR patches to that of HR patches, following the manifold assumption already used in the earlier work of Chang et al. [11]. The mapping of the manifold is assumed to be locally linear and therefore several linear regressors are used and anchored to the manifold as a piecewise linearization. Although these methods are among the fastest in the state-of-the-art, searching the proper regressor takes a significant quota of the running time from within the whole SR pipeline.

In this paper we introduce the following contributions:

1. We propose a training scheme which noticeably improves the quality of linear regression for SR (Section 3.1) while keeping the same testing complexity, i.e. not increasing testing time.
2. We formulate an inverse-search approach where for every regressor in the dictionary its k-Nearest Neighbors (k-NN) input image features are found (Fig. 1). Also, we provide a suitable and efficient spherical hashing framework to exploit this scheme, which greatly improves speed at little quality cost. (Section 3.2).

By merging the two contributions, we improve both in speed and quality to both the fastest and the best-performing state-of-the-art methods, as shown in the experimental results.

2 Regression-based SR

In this section we introduce the SR problem and how example-based approaches tackle it, followed by a review of the recent state-of-the-art regression work of Timofte et al. [12] as it is closely related with the work presented in this paper. The contributions of this paper follow in Section 3.

2.1 Problem statement

Super-Resolution aims to upscale images which have an unsatisfactory pixel resolution while preserving the same visual sharpness, more formally

$$X = \uparrow (Y) \; s.t. \; \mathcal{X} \approx \mathcal{Y}, \tag{1}$$

where Y is the input image, X is the output upscaled image, $\uparrow (\cdot)$ is an upsampling operator and calligraphic font denotes the spectrum of an image.

In the literature this transformation has usually been modeled backwards as the restoration of an original image that has suffered several degradations [9]

$$Y = \downarrow (B(X)), \tag{2}$$

where $B(\cdot)$ is a blurring filter and $\downarrow(\cdot)$ is a downsampling operator. The problem is usually addressed at a patch level, denoted with lower case (e.g. y, x).

The example-based SR family tackles the super-resolution problem by finding meaningful examples from which a HR counterpart is already known, namely the couple of dictionaries D_l and D_h:

$$\min_{\beta} \|y - D_l\beta\|_2^2 + \lambda \|\beta\|_p , \tag{3}$$

where β selects and weights the elements in the dictionary and λ weights a possible L_p-norm regularization term. The L_p-norm selection and the dictionary-building process depend on the chosen priors and they further define the SR algorithm.

2.2 Anchored Neighborhood Regression

The recent work of Timofte et al. [12] is especially remarkable for its low-complexity nature which achieves orders of magnitude speed-ups while having competitive quality results compared to the state-of-the-art. They proposed a relaxation of the L_1-norm regularization commonly used in most of the Neighbor Embedding (NE) and Sparse Coding (SC) approaches, reformulating the problem as a least squares (LS) L_2-norm regularized regression, also known as Ridge Regression. While solving L_1-norm constrained minimization problems is computationally demanding, when relaxing it to a L_2-norm, a closed-form solution can be used. Their proposed minimization problem reads

$$\min_{\beta} \|y_F - N_l\beta\|_2^2 + \lambda \|\beta\|_2 , \tag{4}$$

where N_l is the LR neighborhood chosen to solve the problem and y_F is a feature extracted from a LR patch. The algebraic solution is

$$\beta = (N_l^T N_l + \lambda I)^{-1} N_l^T y_F. \tag{5}$$

The coefficients of β are applied to the corresponding HR neighborhood N_h to reconstruct the HR patch, i.e. $x = N_h\beta$. This can also be written as the matrix multiplication $x = R\, y_F$, where the projection matrix (i.e. regressor) is calculated as

$$R = N_h(N_l^T N_l + \lambda I)^{-1} N_l^T \tag{6}$$

and can be computed offline, therefore moving the minimization problem from testing to training time.

They propose to use sparse dictionaries of d_s atoms size, trained with the K-SVD algorithm [13]. A regressor R_j is anchored to each atom d_j in D_l, and the neighborhood N_l in equation (6) is selected from a k-NN subset of D_l:

$$N_{l_j} = \text{kNN}(d_j,\, D_l). \tag{7}$$

The SR problem can be addressed by finding the NN atom d_j of every input patch feature y_{iF} and applying the associated R_j to it. In the specific case of a neighborhood size $k = d_s$, only one general regressor is obtained whose neighborhood comprises all the atoms of the dictionary and consequently does not require a NN search. This case is referred in the original paper as Global Regression (GR).

2.3 Linear regression framework

Once the closest anchor point is found, the regression is usually applied to certain input features and aims to recover certain components of the patch. We model the linear regression framework in a general way as

$$x = \tilde{x} + R\, y_F, \tag{8}$$

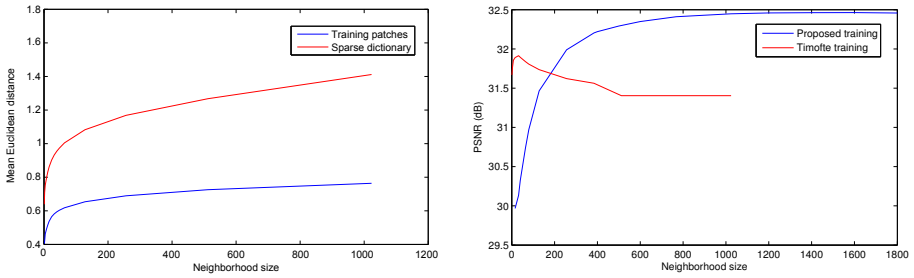
where $\tilde{x}$ is a coarse first approximation of the HR patch x. The choice of how to obtain $\tilde{x}$ requires selecting a prior on how to better approximate x. In the work of Yang et al. [8] they use the in-place prior as this first-approximation and Timofte et al. [12] use the bicubic interpolation assuming a smooth prior. The regressors are trained to improve the reconstruction whenever the coarse prior is not sufficient. Intuitively, for an optimal performance, the selected feature representation has to be related with the chosen first approximation $\tilde{x}$. Supporting this, [8] uses as input feature the subtraction of the low-pass filtered in-place example to the bicubic interpolation, intuitively modeling the errors of the in-place prior for the low-frequency band; and [12] uses gradient-based features, representing the high-frequency components which are likely not going to be well-reconstructed with bicubic interpolation.

3 Fast hashing-based Super-resolution

In this section we present our super-resolution algorithm based on an inverse-search scheme. The section is divided into two parts representing the contributions of the paper: We first discuss the optimal training stage for linear super-resolution regressors and then introduce our hashing-based regressor selection scheme.

3.1 Training

In regression-based SR the objective of training a given regressor R is to obtain a certain mapping function from LR to HR patches. From a more general perspective, LR patches form an input manifold M of dimension m and HR patches form a target manifold N of dimension n. Formally, for training pairs $(y_{Fi},\, x_i)$ with $y_F \in M$ and $x_i \in N$, we would like to infer a mapping $\Psi : M \subseteq \mathbb{R}^m \rightarrow N \subseteq \mathbb{R}^n$.



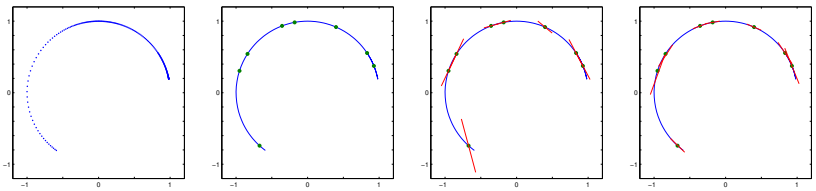
(a) (b)

Fig. 2: (a) Mean euclidean distance between atoms and its neighborhood for different neighborhood sizes. (b) Quality improvement measured in PSNR (dB) for a reconstruction using ANR [12] together with our proposed training. 1024 anchor points were used for this experiment.

As we have previously seen, recent regression-based SR use linear regressors because they can be easily computed in closed form and applied as a matrix multiplication. However, the mapping Ψ is highly complex and non-linear [14]. To model the non-linearity nature of the mapping, an ensemble of regressors $\{R_i\}$ is trained, representing a locally linear parametrization of Ψ, under the assumption that both manifolds M and N have a similar local geometry. We analyze the effect on the distribution of those regressors in the manifold (i.e. the anchor points) and the importance of properly choosing the N_l in equation (6), concluding on a new training approach.

In the work of Timofte et al. [12], an overcomplete sparse representation is obtained from the initial LR training patches using K-SVD [13]. This new reduced dictionary D_l is used both as anchor points to the manifold and datapoints for the regression training. In their GR, a unique regressor R_G is trained with all elements of the dictionary, therefore accepting higher regression errors due to the single linearization of the manifold. For a more fine-tuned regression reconstruction they also propose the Anchored Neighborhood Regression (ANR), they use as anchor points $\{A_1, \ldots, A_{d_s}\}$ the dictionary points $\{D_1, \ldots, D_{d_s}\}$ and they build for each one of those atoms a neighborhood of k-NN within the same sparse dictionary D_l.

Performing a sparse decomposition of a high number of patches efficiently compresses data in a much smaller dictionary, yielding atoms which are representative of the whole training dataset, i.e. the whole manifold. For this reason they are suitable to be used as anchor points, but also sub-optimal for the neighborhood embedding. They are sub-optimal since the necessary local condition for the linearity assumption is likely to be violated. Due to the L_1-norm reconstruction minimization imposed in sparse dictionaries, atoms in the dictionary are not close in the Euclidean space, as shown in Fig. 2(a).



(a) (b) (c) (d)

Fig. 3: A normalized degree 3 polynomial manifold illustrating the proposed approach compared to the one in [12]. (a) Bidimensional manifold samples. (b) The manifold (blue) and the sparse representation obtained with K-SVD algorithm (green) of 8 atoms. (c) Linear regressors (red) trained with the neighborhoods ($k = 1$) obtained within the sparse dictionary, as in [12]. (d) Linear regressors (red) obtained using our proposed approach: The neighborhoods are obtained within the samples from the manifold ($k = 10$).

This observation leads us to propose a different approach when training linear regressors for SR: Using sparse representations as anchor points to the manifold, but forming the neighborhoods with raw manifold samples (e.g. features, patches). In Fig. 2(a) we show how, by doing so, we find closer nearest neighbors and, therefore, fulfill better the local condition. Additionally, a higher number of local independent measurements is available (e.g. mean distance for 1000 neighbors in the raw-patch approach is comparable to a 40 atom neighborhood in the sparse approach) and we can control the number of k-NN selected, i.e. it is not upper-bounded by the dictionary size. We show a low-dimensional example of our proposed training scheme in Fig. 3.

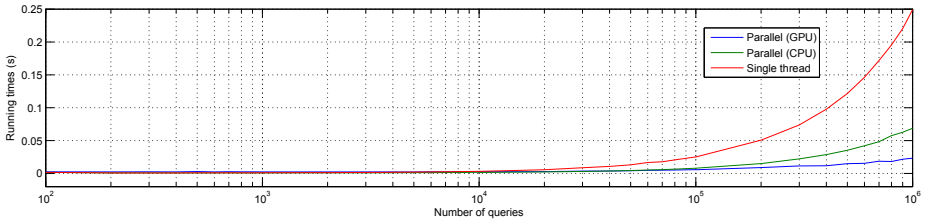


Fig. 4: Running times measured when computing 6-bit hash codes (6 hyperspheres) for increasing number of queries (in logarithmic axis and without reranking) for single-threaded (CPU) and parallel (CPU and GPU) implementations.

In Fig. 2(b) we show the comparison of ANR [12] with both training approaches in terms of resulting image PSNR. We use the same training dataset of the original paper, and for our neighbor embedding we use the L_2-normalized

raw features which are introduced in the K-SVD algorithm. We fix the dictionary size (only used as anchor points in our scheme) to 1024. By applying our training scheme we achieve substantial quality improvements, both qualitatively and quantitatively.

3.2 Inverse search via spherical hashing

When aiming at a fine modeling of the mapping between LR and HR manifolds, several linear regressors have to be trained to better represent the non-linear problem. Although state-of-the art regression-based SR has already pushed forward the computational speed with regard to other dictionary-based SR [10, 9], finding the right regressor for each patch is still consuming most of the execution time. In the work of [12], most of the *encoding time* (i.e. time left after subtracting shared processing time, including bicubic interpolations, patch extractions, etc.) is spent in this task (i.e. $\sim 96\%$ of the time).

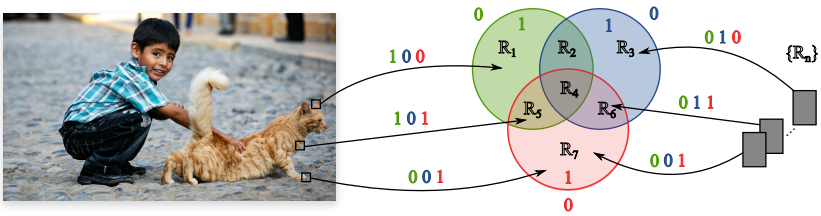


Fig. 5: Spherical hashing applied for the inverse search super-resolution problem. Certain hashing functions are optimized on feature patch statistics creating a set of hyperspheres intersections that are directly labeled with a hash code. In training time, regressors fill this intersections (i.e. bins) and in testing time the hashing function is applied to each patch, which will directly map it to a regressor.

The second contribution of this paper is a novel search strategy designed to benefit from the training outcome presented in Section 3.1, i.e. anchor points of the dictionary and their neighborhoods are obtained independently and ahead from the search structure.

In order to improve the search efficiency, search structures of sublinear complexity are often built, usually in the form of binary splits, e.g. trees, hashing schemes [15–18]. One might consider determining the search partitions with the set of anchor points, since those are the elements to retrieve. However, the small cardinality of this set leads to an imprecise partitioning due to a shortage of sampling density. We propose an inverse search scheme which consists in finding the k-ANN (Approximate Nearest Neighbor) patches within the image for every anchor point, as shown in Fig. 1. By doing so, we have a dense sampling (i.e. all training patches) at our disposal, which results in meaningful partitions.

We choose hashing techniques over tree-based methods. Hashing schemes provide low memory usage (the number of splitting functions in hashing-based structures is $O(\log_2(n))$ while in tree-based structures is $O(n)$, where n represents the number of clusters) and are highly parallelizable.

Binary hashing techniques aim to embed high-dimensional points in binary codes, providing a compact representation of high-dimensional data. Among their vast range of applications, they can be used for efficient similarity search, including approximate nearest neighbor retrieval, since hashing codes preserve relative distances. There has recently been active research in data-dependent hashing functions opposed to hashing methods such as [17] which are data-independent. Data-dependent methods intend to better fit the hashing function to the data distribution [19, 18] through an off-line training stage.

Among the data-dependent state-of-the-arts methods, we select the Spherical Hashing algorithm of Heo et al. [16], which is able to define closed regions in $\mathbb{R}^m$ with as few as one splitting function. This hashing framework is useful to model the inverse search scheme and enables to benefit from substantial speed-ups by reducing the NN search into applying a precomputed function, which conveniently scales with parallel implementations, as shown in Fig. 4.

Spherical hashing differs from previous approaches by setting hyperspheres to define hashing functions on behalf of the previously used hyperplanes. A given hashing function $H(y_F) = (h_1(y_F), \ldots, h_c(y_F))$ maps points from $\mathbb{R}^m$ to a base 2 $\mathbb{N}^c$, i.e. $\{0,1\}^c$. Every hashing function $h_k(y_F)$ indicates whether the point y_F is inside kth hypersphere, modeled for this purpose as a *pivot* $p_k \in \mathbb{R}^m$ and a distance threshold (i.e. radius of the hypersphere) $t_k \in \mathbb{R}^+$ as:

$$h_k(y_F) = \begin{cases} 0 & when\ d(p_k, y_F) > t_k \\ 1 & when\ d(d_k, y_F) \leq t_k \end{cases}, \tag{9}$$

where $d(p_k, y_F)$ denotes a distance metric (e.g. Euclidean distance) between two points in $\mathbb{R}^m$. The advantages of using hyperspheres instead of hyperplanes is the ability to define closed tighter sub-spaces in $\mathbb{R}^m$ as intersection of hyperspheres. An iterative optimization training process is proposed in [16] to obtain the set $\{p_k, t_k\}$, aiming a balanced partitioning of the training data and independence between hashing functions.

We perform this mentioned iterative hashing-function optimization in a set of input patch features from training images, so that $H(y_F)$ adapts to the natural image distribution in the feature space. Our proposed spherical hashing search scheme becomes symmetrical as we can see in Fig. 5, i.e. both image and anchor points have to be labeled with binary codes. This can be intuitively understood as creating NN subspace groups (we refer them as *bins*), which we label with a regressor by applying the same hashing functions to the anchor points. Relating a hash code with a regressor can be done during training time.

The inverse search approach returns k-NN for each anchor point, thus not ensuring that all the input image patches have a related regressor (i.e. whenever the patch is not within the k-NN of any of the anchor points). Two solutions are proposed: (a) use a general regressor for the patches which are not in the k-NN of

any anchor point or (b) use the regressor of the closest labeled hash code calculated with the spherical Hamming distance, defined by [16] as $d_{SH}(a,b) = \frac{\sum(a \oplus b)}{\sum(a \wedge b)}$, where $\oplus$ is the XOR bit operation and $\wedge$ is the AND bit operation. Note that although not being guaranteed, it rarely happens that a patch is not within any of the k-NN regressors (e.g. for the selected parameter of 6 hyperspheres it never occurs). Since we have not observed significant differences in performance, we select (a) as the lowest complexity solution, although more testing on (b) is due.

In a similar way, an inverse search might also assign two or more regressors to a single patch. It is common in the literature to do a re-ranking strategy to deal with this issue [20].

Table 1: Performance of $\times 3$ and $\times 4$ magnification in terms of averaged PSNR (dB) and averaged execution time (s) on datasets Set14, Kodak and 2k.

		Bicubic		Sparse [10]		GR [12]		ANR [12]		NE+LS		NE+NNLS		NE+LLE		Proposed	
	MF	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
Set14	3	27.54	0.002	28.67	2.981	28.31	0.528	28.65	0.771	28.59	2.854	28.44	25.372	28.60	4.356	**28.93**	**0.188**
	4	26.00	0.003	26.88	1.862	26.60	0.458	26.85	0.584	26.81	1.716	26.72	14.146	26.81	2.623	**27.04**	**0.184**
Kodak	3	28.43	0.003	29.22	5.126	28.98	0.921	29.21	1.335	29.17	4.829	29.04	44.102	29.17	7.353	**29.42**	**0.314**
	4	27.23	0.003	27.83	3.194	27.64	0.757	27.80	1.022	27.77	3.003	27.71	24.428	27.77	4.678	**27.92**	**0.309**
2k	3	31.73	0.007	32.63	27.622	32.45	4.860	32.68	7.123	32.62	26.194	32.51	242.875	32.65	40.389	**32.88**	**1.652**
	4	30.28	0.006	30.97	17.225	30.81	3.968	30.99	5.344	30.94	16.363	30.87	136.058	30.96	25.967	**31.04**	**1.578**

4 Results

In this section we show experimental results of our proposed method and we compare its performance in terms of quality and execution time to other state-of-the-art recent methods. We perform extensive experiments with image resolutions ranging from 2.5Kpixels to 2Mpixels, showing the performance for classic literature testing images but additionally demonstrating how these algorithms would perform in current upscaling scenarios. We further extend the benchmark in [12] by adding to *Set5* and *Set14* two more datasets: the 24 image *kodak* dataset and *2k*, which is a image set of 9 sharp images obtained from the internet with a pixel resolution of 1920x1080.

All the experiments were run on a Intel Xeon W3690 @ 3.47GHz and the code of the compared methods was obtained from [12] and used with their recommended parameters. The methods compared are the sparse coding SR of Zeyde et al [10], an implementation of the LS regressions used by Chang et al. [11] (NE + LLE), and the Non-Negative Least Squares (NE+NNLS) method of Bevilaqcua et al. [21].

The proposed algorithm is written in MATLAB with the most time-consuming stages implemented in OpenCL without further emphasis in optimization, and

Table 2: Performance of ×3 and ×4 magnification in terms of PSNR (dB) and execution time (s) on the Set5 dataset.

Set5		Bicubic		Sparse [10]		GR [12]		ANR [12]		NE+LS		NE+NNLS		NE+LLE		Proposed	
images	MF	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
baby	3	33.9	0.000	35.1	3.490	34.9	0.662	35.1	0.905	35.0	3.179	34.8	29.377	35.1	5.042	35.1	0.214
bird	3	32.6	0.000	34.6	1.087	33.9	0.242	34.6	0.293	34.4	1.011	34.3	9.449	34.6	1.533	34.9	0.070
butterfly	3	24.0	0.000	25.9	0.839	25.0	0.152	25.9	0.201	25.8	0.766	25.6	6.947	25.8	1.200	26.6	0.058
head	3	32.9	0.000	33.6	1.011	33.5	0.218	33.6	0.270	33.5	0.908	33.5	8.411	33.6	1.395	33.7	0.068
woman	3	28.6	0.000	30.4	0.972	29.7	0.187	30.3	0.249	30.2	0.909	29.9	8.437	30.2	1.390	30.06	0.067
average	3	30.39	0.000	31.90	1.480	31.41	0.292	31.92	0.384	31.78	1.354	31.60	12.524	31.84	2.112	**32.22**	**0.095**
baby	4	31.8	0.000	33.1	2.136	32.8	0.525	33.0	0.652	32.9	2.033	32.8	15.535	33.0	3.128	32.9	0.256
bird	4	30.2	0.000	31.7	0.660	31.3	0.184	31.8	0.226	31.6	0.611	31.5	4.995	31.7	0.955	31.7	0.066
butterfly	4	22.1	0.000	23.6	0.536	23.1	0.138	23.5	0.165	23.4	0.456	23.3	3.882	23.4	0.730	23.7	0.052
head	4	31.6	0.000	32.2	0.582	32.1	0.135	32.3	0.212	32.2	0.567	32.1	4.587	32.2	0.882	32.3	0.061
woman	4	26.5	0.000	27.9	0.576	27.4	0.174	27.8	0.191	27.6	0.583	27.6	4.455	27.7	0.894	28.0	0.063
average	4	28.42	0.000	29.69	0.898	29.34	0.231	29.69	0.289	29.55	0.850	29.47	6,691	29,61	1.318	**29.73**	**0.100**

runs in the same CPU platform used for all methods. Our experiments use the same K-SVD sparse dictionary of 1024 used for the compared methods.

We selected bicubic as our coarse approximation $\tilde{x}$ since it does not limit the upscaling steps for super-resolution (e.g. in-place examples are only meaningful for very small magnification factors) and also the features used by Zeyde et al. [10, 12] composed by 1st and 2nd order derivative filters compressed with PCA and truncating when the feature still conserves 99.9% of its energy. We also use a L_2-norm regularized linear regressor illustrated in equation (4). We build therefore on top of the regressor scheme proposed by Timofte et al. [12]. We used 6-bit spherical hashing (6 hyperspheres) and the chosen neighborhood is of 1300 k-NN. The selection of number of spheres is a trade-off between quality and speed, since when we decrease the number of hyperspheres we have more collision of regressors (i.e. more than one regressor arrives to the same bin) and due to the re-ranking process we get closer to an exact nearest neighbor search. This can be seen in Fig. 7.

In Table 1 and Table 2 we show objective results of the performance in terms of PSNR (dB) and execution time (s). For both measures, our proposed algorithm is the best performing. The improvement in PSNR is more noticeable for magnification factors of 3, where we reach improvements of up to 0.3 dB when compared to the second best-performer. In terms of running time, our algorithm has consistent speed-ups in all datasets and all scales. When compared to GR (which is the fastest of the compared methods), the speed-ups are ranging from ×2 to ×3 , additionally with a gap in quality reconstruction. The speed-ups for ANR range from ×3 to ×4 and for the rest of the methods, the running times are several orders of magnitudes slower. Note that the theoretical complexity of GR is lower than that of our method since it does not perform a NN search (i.e. for similar implementations, GR should be slightly faster). Nevertheless, our parallel implementation is more efficient than the one provided by their authors [12], which mostly relies on optimized MATLAB matrix multiplication.

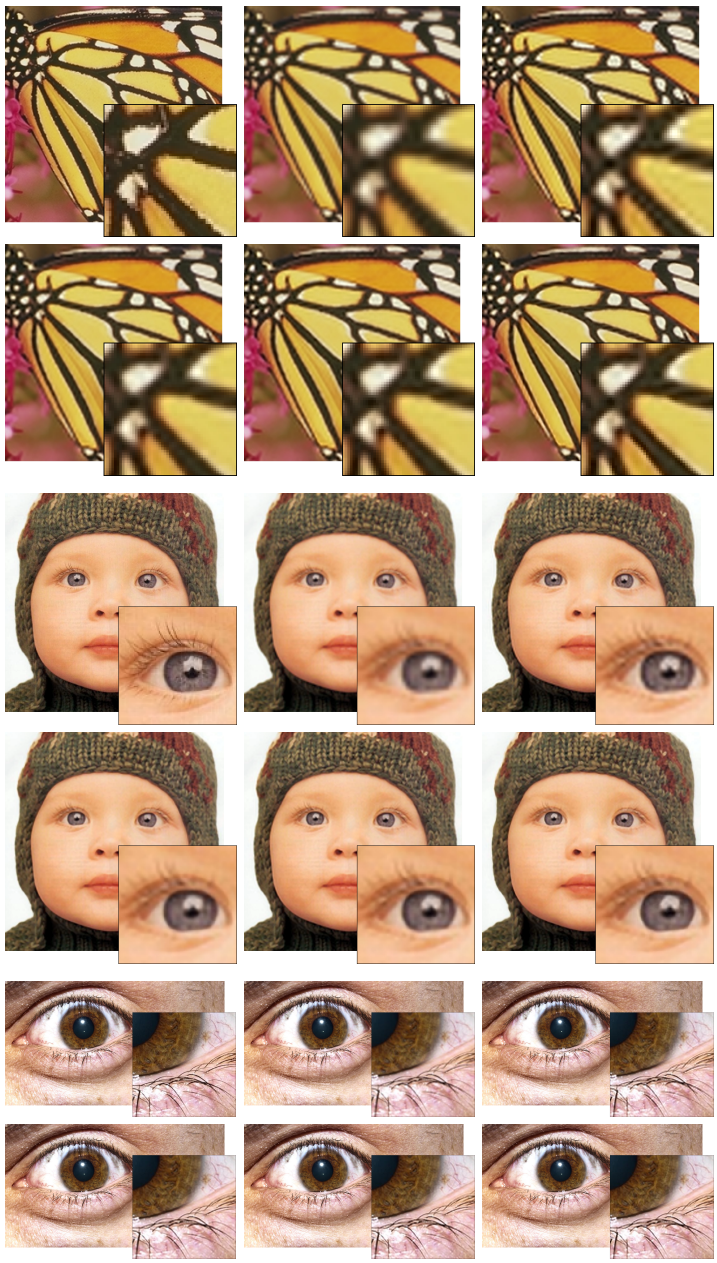
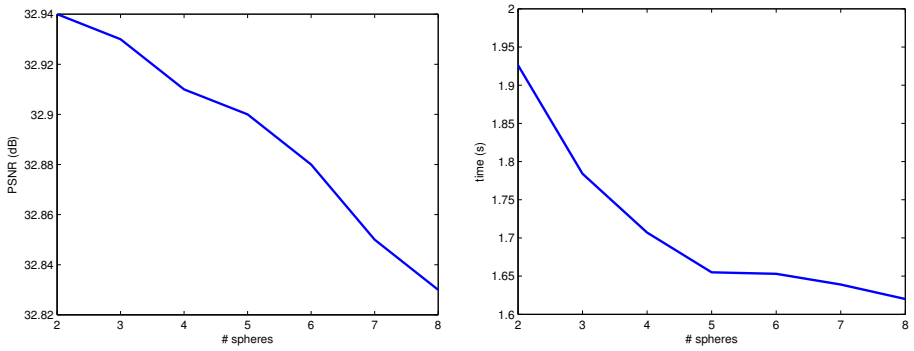


Fig. 6: Visual qualitative assessment of $\times 3$ magnification factor for images from different datasets. From left to right and top to bottom: Original, bicubic interpolation, Global Regressor [12], Zeyde et al. [10], ANR [12] and Proposed SR. Better viewed zoomed in.



(a) (b)

Fig. 7: Effect of the number of spheres selected in terms of PSNR (a) and time (b).

In Fig. 6 a visual qualitative assessment can be performed. Our method obtains more natural and sharp edges, and strongly reduces ringing. A good example of that is shown in the *butterfly* image.

5 Conclusions

In this paper we have presented two main contributions: An improved training stage and an efficient inverse-search approach for regression-based and, more generally, dictionary-based SR. Spherical hashing techniques have been applied in order to exploit the benefits of the inverse-search scheme. We obtain both quality improvements due to the optimal training stage and also substantial speed-ups from the low-complexity spherical hashing similarity algorithm used in the regressor selection. An exhaustive testing has been performed comparing our method with four datasets of several pixel resolutions, with different upscaling factors and with several state-of-the-art methods. Our experimental results show consistent improvements in PSNR and running times over the state-of-the-art methods included in the benchmark, positioning as the first in both measures.

References

1. Tsai, R., Huang, T.: Multiple frame image restoration and registration. In: Proc. Advances in Computer Vision and Image Processing. Volume 1. (1984) 317–339
2. Irani, M., Peleg, S.: Improving resolution by image registration. CVGIP: Graphical Models and Image Processing **53** (1991) 231–239
3. Baker, S., Kanade, T.: Limits on super-resolution and how to break them. IEEE Trans. on Pattern Analysis and Machine Intelligence **24** (2002) 1167–1183

4. Lin, Z., Shum, H.Y.: Fundamental limits of reconstruction-based superresolution algorithms under local translation. IEEE Trans. on Pattern Analysis and Machine Intelligence **26** (2004) 83–97
5. Freeman, W., Jones, T., Pasztor, E.: Example-based super-resolution. IEEE Trans. Computer Graphics and Applications **22** (2002) 56–65
6. Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: Proc. IEEE Int. Conference on Computer Vision. (2009)
7. Freedman, G., Fattal, R.: Image and video upscaling from local self-examples. ACM Trans. on Graphics **30** (2011) 12:1–12:11
8. Yang, J., Lin, Z., Cohen, S.: Fast image super-resolution based on in-place example regression. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition. (2013)
9. Yang, J., Wright, J., T.S., H., Ma, Y.: Image super-resolution via sparse representation. IEEE Trans. on Image Processing **19** (2010) 2861–2873
10. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: Proc. Int. Conference on Curves and Surfaces. (2012)
11. Chang, H., Yeung, D.Y., Xiong, Y.: Super-resolution through neighbor embedding. (2004)
12. Timofte, R., Smet, V.D., Goool, L.V.: Anchored neighborhood regression for fast example-based super-resolution. In: Proc. IEEE Int. Conference on Computer Vision. (2013)
13. Aharon, M., Elad, M., Bruckstein, A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Trans. on Signal Processing **54** (2006)
14. Peyré, G.: Manifold models for signals and images. Computer Vision and Image Understanding **113** (2009) 249–260
15. Breiman, L.: Random forests. Machine Learning **45** (2001) 5–32
16. Heo, J.P., Lee, Y., He, J., Chang, S.F., Yoon, S.E.: Spherical hashing. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition. (2012)
17. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Proc. of the Thirtieth Annual ACM Symposium on Theory of Computing. STOC '98 (1998) 604–613
18. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for scalable image retrieval. (2010)
19. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. (2008)
20. He, K., Sun, J.: Computing nearest-neighbor fields via propagation-assisted kd-trees. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition. (2012)
21. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.L.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In: Proc. of the British Machine Vision Conf. (2012) 1–10