Single-Image Super-resolution Using Sparse Regression and Natural Image Prior

Kwang In Kim and Younghee Kwon

Abstract—This paper proposes a framework for single-image super-resolution. The underlying idea is to learn a map from input low-resolution images to target high-resolution images based on example pairs of input and output images. Kernel ridge regression (KRR) is adopted for this purpose. To reduce the time complexity of training and testing for KRR, a sparse solution is found by combining the ideas of kernel matching pursuit and gradient descent. As a regularized solution, KRR leads to a better generalization than simply storing the examples as it has been done in existing example-based algorithms and results in much less noisy images. However, this may introduce blurring and ringing artifacts around major edges as sharp changes are penalized severely. A prior model of a generic image class which takes into account the discontinuity property of images is adopted to resolve this problem. Comparison with existing algorithms shows the effectiveness of the proposed method.

Index Terms—Computer vision, machine learning, image enhancement, display algorithms

1 Introduction

Single-image super-resolution refers to the task of constructing a high-resolution enlargement of a *single* low-resolution image. This problem is inherently ill-posed as there are generally multiple high-resolution images that can be reduced to the same low-resolution image. Accordingly, for this problem, one has to rely on strong prior information. This information is available either in the explicit form of a distribution or energy functional defined on the image class [1], [2], [3], [4], and/or in the implicit form of example images which leads to example-based super-resolution [5], [6], [7], [8], [9], [10], [11], [12], [13].

Previous example-based super-resolution algorithms can roughly be characterized as nearest neighbor (NN)-based estimation: during the *training phase*, pairs of low-resolution and corresponding high-resolution image patches (sub-windows of images) are collected. Then, in the *super-resolution phase*, each patch of the given low-resolution image is compared to the stored low-resolution patches, and the high-resolution patch corresponding to the nearest low-resolution patch and satisfying a certain spatial neighborhood compatibility is selected as the output. For instance, Freeman et al. [6] posed the image super-resolution as the problem of estimating high-frequency details by interpolating the input low-resolution image into the desired scale (which results in a blurred image). Then, the super-resolution is performed by the NN-based estimation of high-frequency patches based on the corresponding patches of input low-frequency image and resolving the compatibility of output patches using a Markov network.

Although this method (and also other NN-based methods) has already shown impressive performance, there is still room for improvement if one views the image super-resolution as a regression problem, i.e., finding a map from the space of low-resolution image patches to the space of target high-resolution patches. It is well known in the machine learning community that NN-based estimation suffers from *overfitting* when the target function is highly complex or the data is high-dimensional [14], which is the case for image super-resolution. Accordingly, it is reasonable to expect that NN-based methods can be improved by adopting learning algorithms with *regularization* capability to avoid overfitting.

Indeed, attempts have already been made to regularize the estimator. Chang et al. [12] regularized the NN estimator by representing the input and target image patches with linear combinations (calculated from locally linear embedding) of stored training patches (k-NNs) while Datsenko and Elad [13] proposed a *maximum a posteriori* (MAP) framework where the prior penalizes the deviation of the solution from a weighted average of k-NNs. The weights are then chosen in a manner similar to robust regression such that the contributions of the outliers are weakened.

A rather straightforward approach would be to regularize the regressor directly. Based on the framework of Freeman et al. [6], [7], Kim et al. [15] have posed the problem of estimating the high-frequency details as a regression problem which is then resolved by support vector regression (SVR). Meanwhile, Ni and Nguyen [16] utilized SVR in the frequency domain and posed the super-resolution as a kernel learning problem. While SVR produced a significant improvement over existing example-based methods, it has drawbacks in building a practical system: 1. as a regularization framework, SVR tends to smooth sharp edges and produce oscillations along the major edges (ringing artifacts). These might lead to low reconstruction error on average, but is visually implausible; 2. SVR results in a dense solution, i.e., the regression function is expanded in the whole set of training data points and accordingly is computationally demanding both in training and in testing: optimizing the hyper-parameters based on cross-validation indicated that the optimum value of ϵ for the ϵ-*insensitive loss function* of SVR is close to zero [15].

The current work extends the framework of Kim et al. [15].[1] Kernel ridge regression (KRR) is utilized in place of SVR. Since the L^2-loss adopted by KRR is differentiable, we construct the sparse basis set based on the combination of the kernel matching pursuit (KMP) [18] and gradient descent, and thereby reduce the time complexity of training and testing for regression. As the regularizer of KRR is the same as that of SVR, the problem of ringing artifacts still remains. This is resolved by exploiting a prior over image structure which takes into account the discontinuity of pixel values across edges.

2 Learning Image Super-resolution

Adopting the framework of Freeman et al. [6], [7], for the super-resolution of a given image, we firstly interpolate the input into the desired scale using cubic spline interpolation (henceforth referred to as 'interpolation'). Then, the high-frequency details which are missing in the interpolation (X) are estimated based on its band frequency components (LX)

- *K. I. Kim is with Max-Planck-Institut für biologische Kybernetik, Spemannstr. 38, D-72076 Tübingen, Germany.*
  E-mail: kimki@tuebingen.mpg.de
- *Y. Kwon is with KAIST, 373-1 Gusong-dong, Yuseong-gu, Daejeon, Korea.*
  E-mail: kyhee@ai.kaist.ac.kr

[1] A short version of this paper appeared in the proceedings of DAGM2008 [17].

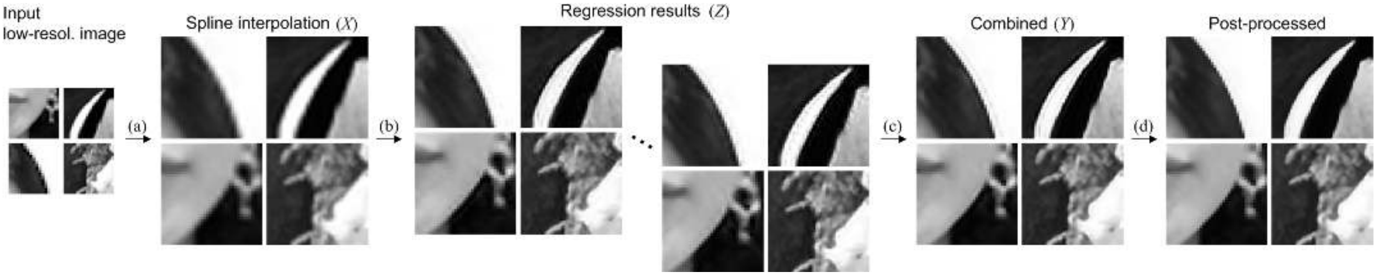


Fig. 1. Overview of super-resolution shown with examples: (a) input image is interpolated into the desired scale, (b) a set of candidate images is generated as the result of regression, (c) candidates are combined based on estimated confidences; The combined result is sharper and less noisy than individual candidates, which however shows ringing artifacts, and (d) post-processing removes ringing artifacts and further enhances edges.

extracted by applying the Laplacian to X. The estimate (Y) can then be added to X to produce the super-resolved image.

A local patch-based regression (cf. Sect. 2.1) is adopted for the estimation: LX is scanned with a patch (of size M, or $\sqrt{M} \times \sqrt{M}$) to produce a patch-valued regression result (of size N) for each pixel. As the output patches overlap with their neighbors, this results in a set of candidates for each pixel location which constitutes a 3-D image Z. Each candidate is obtained based on different local observation of input image and accordingly contains different partial information of the underlying high-resolution image. A single high-resolution image is then obtained as a convex combination for each pixel of the set of candidate pixels based on their estimated confidence. To enhance the visual quality around the major edges, the results are post-processed based on the prior of natural images proposed by Tappen et al. [1] (cf. Sect. 2.2). Figure 1 summarizes the super-resolution process.

2.1 Regression and Combination

The training patch pairs for the regression are randomly sampled from a set of low-resolution and corresponding desired high-resolution images (cf. Sect. 3). To avoid that the learning is distracted by uninformative patterns, the patches whose norms are close to zero are excluded from the training set. Furthermore, to increase the efficiency of the training set, the data are contrast-normalized [7]: during the construction of the training set, both the input patch and corresponding desired patches are normalized by dividing them by the L^1-norm of the input patch. For an unseen image patch, the input is again normalized before the regression and the corresponding output is inverse normalized.

For a given set of training data points $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_l, \mathbf{y}_l)\} \subset \mathbb{R}^M \times \mathbb{R}^N$, we minimize the following regularized cost functional for the regressor $\mathbf{f} = \{f^1, \ldots, f^N\}$:

$$\mathcal{E}(\mathbf{f}) = \frac{1}{2} \sum_{i=1,\ldots,N} \left(\sum_{j=1,\ldots,l} (f^i(\mathbf{x}_j) - y^i_j)^2 + \lambda^i \|f^i\|^2_{\mathcal{H}^i} \right), \quad (1)$$

where $\mathbf{y}_j = [y^1_j, \ldots, y^N_j]^\top$ and $\mathcal{H}^i$ is a *reproducing kernel Hilbert space* (RKHS). Due to the *reproducing property* (i.e. $\langle f, k(x, \cdot)\rangle_{\mathcal{H}} = f(x)$), the minimizer of above functional is expanded in kernel functions:

$$f^i(\cdot) = \sum_{j=1,\ldots,l} a^i_j k^i(\mathbf{x}_j, \cdot), \text{ for } i = 1, \ldots, N,$$

and

$$\|f^i\|^2_{\mathcal{H}^i} = \sum_{m,n=1,\ldots,l} a^i_m a^i_n k^i(\mathbf{x}_m, \mathbf{x}_n), \text{ for } i = 1, \ldots, N,$$

where k^i is the *reproducing kernel* [19] for $\mathcal{H}^i$, e.g., to be a Gaussian kernel

$$k^i(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{\sigma^i_k}\right).$$

Equation (1) is the sum of individual convex cost functionals for each scalar-valued regressor f^i and the minimum can obtained analytically. However, this requires the construction and inversion of N kernel matrices ($[\mathbf{K}^i_{(m,n)}]_{l,l} = k^i(\mathbf{x}_m, \mathbf{x}_n)$, for $i = 1, \ldots, N$) in training and $N \times l$ kernel evaluations in testing, which becomes prohibitive even for a relatively small number of training data points (e.g., $l \approx 10,000$) (cf. [17] for details).

In this paper, this problem is approached by trading the complexity off with the optimality of the solution by 1. tying the regularization parameter and the kernel parameter for the regressors (i.e. $\lambda = \lambda^i$ and $\sigma_k = \sigma^i_k$ for $i = 1, \ldots, N$) and 2. finding the minimizer of (1) only within the span of a *sparse* basis set $\{k(\mathbf{b}_1, \cdot), \ldots, k(\mathbf{b}_{l_b}, \cdot)\}$ ($l_b \ll l$):

$$f^i(\cdot) = \sum_{j=1,\ldots,l_b} a^i_j k(\mathbf{b}_j, \cdot), \text{ for } i = 1, \ldots, N.$$

In this case, by sharing the evaluations of kernel functions, the time complexity of patch-valued regression reduces down to the case of scalar-valued regression, and eventually, the time complexity of testing becomes $\mathcal{O}(M \times l_b)$. Since the solution is obtained by

$$\mathbf{A} = (\mathbf{K}_{bx}\mathbf{K}^\top_{bx} + \lambda \mathbf{K}_{bb})^{-1} \mathbf{K}_{bx} \mathbf{Y},$$

where $[\mathbf{K}_{bx(m,n)}]_{l_b,l} = k(\mathbf{b}_m, \mathbf{x}_n)$, $[\mathbf{K}_{bb(m,n)}]_{l_b,l_b} = k(\mathbf{b}_m, \mathbf{b}_n)$, $\mathbf{Y} = [\mathbf{y}^\top_1, \ldots, \mathbf{y}^\top_l]^\top$, and $[\mathbf{A}_{(j,i)}]_{l_b,N} = a^i_j$, for a given fixed set of *basis points* $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_{l_b}\}$, the time complexity of training is $\mathcal{O}(l^3_b + l \times l_b \times M)$. In general, the total training time depends on the method of finding $\mathcal{B}$.

Since the cost functional (1) is a differentiable function of the basis points $\mathcal{B}$, it can afford gradient-based optimization as already demonstrated in the context of sparse Gaussian process regression [20]. Assuming that the evaluation of the derivative of k with respect to a basis vector takes $\mathcal{O}(M)$-time, which is the case for a Gaussian kernel ($\frac{\partial}{\partial \mathbf{b}} k(\mathbf{x}, \mathbf{b}) = \frac{2}{\sigma_k} k(\mathbf{x}, \mathbf{b})(\mathbf{x} - \mathbf{b})$), the evaluation of derivatives of (1) with respect to $\mathcal{B}$ and corresponding coefficient matrix $\mathbf{A}$ takes $\mathcal{O}(M \times l \times l_b)$:[2]

[2] With a slight abuse of the Matlab notation, $\mathbf{A}_{(m:n,:)}$ stands for the submatrix of $\mathbf{A}$ obtained by extracting the rows of $\mathbf{A}$ from m to n. Likewise, $\mathbf{A}_{(:,m)}$ is defined as the m-th column of $\mathbf{A}$.

$$\begin{aligned}\frac{\partial}{\partial \mathbf{A}}\mathcal{E}(\mathbf{f}) &= \mathbf{K}_{bx}\left(\mathbf{K}_{bx}^{\top}\mathbf{A}-\mathbf{Y}\right)+\lambda\mathbf{K}_{bb}\mathbf{A}\\ \frac{\partial}{\partial \mathbf{b}_j}\mathcal{E}(\mathbf{f}) &= \frac{\partial \mathbf{K}_{bx(j,:)}}{\partial \mathbf{b}_j}(\mathbf{K}_{bx}^{\top}\mathbf{A}-\mathbf{Y})\mathbf{A}_{(j,:)}^{\top}\\ &\quad +\lambda\frac{\partial \mathbf{K}_{bb(j,:)}}{\partial \mathbf{b}_j}\mathbf{A}\mathbf{A}_{(j,:)}^{\top},\ \text{for } j=1,\ldots,l_b. \quad (2)\end{aligned}$$

However, due to the non-convexity of (1) with respect to $\mathcal{B}$, naïve gradient descent is susceptible to local minima and accordingly a good heuristic is required to initialize the solution.

The KMP is adopted for this purpose. In KMP (with *prefitting*) [18], the basis points are *selected* from the training data points in an incremental way: for given $n-1$ basis points, the n-th basis point is chosen such that the cost functional (1) is minimized when $\mathbf{A}$ is optimized accordingly.[3]

The basic idea is to assume that at the n-th step of KMP, the chosen basis point $\mathbf{b}_n$ plus the accumulation of basis points obtained until the $(n-1)$-th step ($\mathcal{B}_{n-1}$) constitute a good initial search point. Then, at each step of KMP, $\mathcal{B}_n$ can be subsequently optimized by gradient descent. Naïve implementation of this idea is still very expensive. To reduce further the complexity, the following simplifications are adopted: 1. In the KMP step, instead of evaluating the whole training set for choosing $\mathbf{b}_n$, only l_c ($l_c \ll l$) points are considered; 2. Gradient descent of $\mathcal{B}_n$ and corresponding $\mathbf{A}_{(1:n,:)}$ are performed only at the every r-th KMP step. Instead, for each KMP step, only $\mathbf{b}_n$ and $\mathbf{A}_{(n,:)}$ are optimized. In this case, the gradient of (1) with respect to $\mathbf{b}_n$ can be evaluated at $\mathcal{O}(M \times l)$-cost.[4] Furthermore, similarly to [21], for a given $\mathbf{b}_n$ the optimal $\mathbf{A}_{(n,:)}$ can be analytically calculated at the same cost (cf. [17]).

At the n-th step, the l_c-candidate basis points for KMP are selected based on a rather cheap criterion. One approach might be to choose data points which show the largest distances between the corresponding function outputs obtained at the $(n-1)$-th step and the desired training outputs (i.e., to use the training error). However, this might tend to choose outliers as they will show relatively large training errors for regularized regression. To avoid this problem, the neighborhood context of each data point is exploited: we define a cost functional which measures the distance between the current function output and the output of a *localized KRR*

$$\begin{aligned}\mathcal{C}(\mathbf{x}_j) &= \|\mathbf{K}_{bx(1:n,:)}^{\top}\mathbf{A}_{bx(1:n,:)}-\tilde{\mathbf{g}}_j(\mathbf{x}_j)\|^2,\\ &\quad \text{for } j=1,\ldots,l,\end{aligned}$$

where $\tilde{\mathbf{g}}_j = [\tilde{g}_j^1,\ldots,\tilde{g}_j^N]$ is the localized KRR centered at the given input $\mathbf{x}_j$, which is obtained by collecting nearest neighbors (NNs) of $\mathbf{x}_j$ and training the full KRR based on only these NNs. The candidate points are then chosen as the training data points corresponding to the l_c-largest values of $\mathcal{C}$. As a regularization method, the use of localized KRRs can effectively single out the outliers. Furthermore, in the preliminary experiments with 10,000 data points (where it was possible to train the full KRR),[5] it turned out that the outputs of localized KRR on training data points are very close to the *full* KRR outputs: the average squared distance between the outputs of full KRR and localized KRR was less than 1% of the mean squared training error of full KRR. Accordingly, they could be regarded as a rough estimation of full KRR solution which one might have obtained by training on all l data points. However, it should be noted that the localized KRRs cannot be directly applied for regression as they might interpolate poorly on non-training data points.

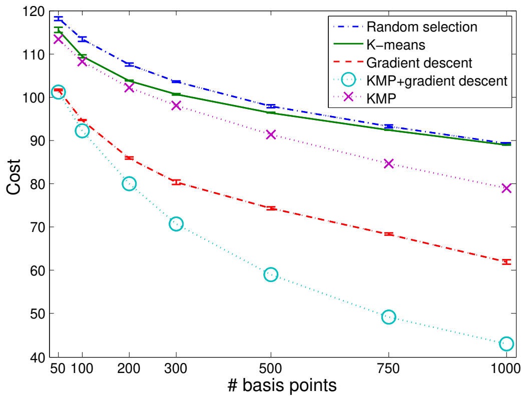


Fig. 2. Performance of different sparse solution methods evaluated in terms of the cost functional (1) for the case of magnification factor 3; A fixed set of hyper-parameters were used for all cases such that the comparison can be made directly in (1). The performance of randomized algorithms (random selection, k-means, gradient descent) are calculated as averages of results from 20 different experiments with random initializations. The lengths of error bars correspond to twice the standard deviations.

To gain an insight into the performance of our basis construction method, a set of experiments has been performed with different sparse solution methods, including random selection (of basis points from the training set), KMP, k-means algorithm (clustering of training data points), naïve gradient descent (with basis initialized by k-means), and the proposed combination of KMP and gradient descent.[6] Fig. 2 summarizes the results. The KMP showed an improved performance over the k-means algorithm and random selection which build the basis set without reflecting the cost functional to be optimized. Both of the two gradient descent methods outperformed KMP which chooses the basis points from the training set. The improved performance of gradient descent in combination with KMP could be attributed to the better initialization of the solution for the subsequent gradient descent step.

As the result of the patch-based regression step, N candidates are generated for each pixel location. This setting is motivated by the observation that 1. by sharing the hyper-parameters and basis points, the computational complexity of patch-valued learning reduces to the scalar-valued learning; 2. the candidates contain information of different input image locations which are actually diverse enough such that the

[3] In the original form of KMP, the regularization was implicitly performed by controlling the number of basis points l_b (i.e., $\lambda = 0$). However, in the current problem, for a given upper bound of l_b, we constantly observed a better generalization performance when we assign l_b with that upper bound and control λ instead.

[4] It should be noted that $[[\mathbf{K}_{bx}]_{n-1,l}^{\top}[\mathbf{A}]_{n-1,N}]_{l,N}$ (cf. (2)) is stored at the $(n-1)$-th step. Accordingly, at the n-th step, augmenting a single row of $\mathbf{K}_{bx}$ and $\mathbf{A}$, respectively is sufficient for calculating the gradient.

[5] For preliminary experiments mentioned in this paper, we used only 10,000 training data points for training the regression part to facilitate fast evaluation.

[6] For this and all the other experiments in this paper, we set the size of interval r and the number of candidate basis points l_c to 10 and 100, respectively.

combination can boost the performance: for the magnification factor 2 case, constructing an image by choosing the best and the worst (in terms of the distance to the ground truth) candidates from each spatial location of Z resulted in an average peak signal-to-noise ratio (PSNR) difference of 7.84dB (cf. the accompanying technical report [22] for details). Certainly, the ground truth is not available at actual super-resolution stage and accordingly a way of constructing a single pixel out of N candidates is required.

In this paper, the final estimation of the pixel value for an image location (x, y) is obtained as the convex combination of candidates given in the form of a *softmax*:[7]

$$Y(x,y) = \sum_{i=1,\dots,N} w_i(x,y) Z(x,y,i),$$

where

$$w_i(x,y) = \exp\left(-\frac{d_i(x,y)}{\sigma_C}\right) \Big/ \sum_{j=1,\dots,N} \exp\left(-\frac{d_j(x,y)}{\sigma_C}\right)$$

and $\{d_1(x,y), \dots, d_N(x,y)\}$ is the estimation of distances between the unknown desired output and each candidate. This estimate is calculated using a set of linear regressors:

$$d_i(x,y) = |PZ(x,y)^\top W_i|, \text{ for } i = 1, \dots, N,$$

where $PZ(x,y)$ is a vector constructed by concatenating all columns of a spatial patch (of size $R \times R \times N$) of Z centered at (x, y) and the parameters $\{W_i\}$ are optimized based on the patch-based regression results (Z) for a subset of training images (cf. Sect. 3).

There are a few hyper-parameters to be tuned: for the regression part, the input and output patch sizes (M and N, respectively), KRR parameters (σ_k and λ), and the number of basis points (l_b) and for the combination part, the input patch size (R) and the weight parameter (σ_C). We fix l_b, N, and R at 300, $25(5 \times 5)$, and $49(7 \times 7)$, respectively. These values are determined by trading the quality of super-resolution off with the computational complexity. We observed constant increase of the performance as l_b increases and becomes larger than 300. Similar tendency was also observed with increasing $N(< M)$ and R while the run-time complexity increases linearly with all these parameters.

The remaining hyper-parameters are chosen based on error rates of super-resolution results for a set of validation images. However, directly optimizing these many parameters is computationally very demanding, especially due to the large time complexity of choosing basis points. With 200,000 training data points, training a sparse KRR for a given fixed parameters took around a day on a 3GHz machine (for the magnification factor 2 case). To retain the complexity of the whole process at a moderate level, we firstly calculate a rough estimation of parameters based on a fixed set of basis points which is obtained from the k-means algorithm. Then, the full validation is performed only at the vicinity of the rough estimation. For the distance measure of k-means clustering, we use the following combination of Euclidean distances from both the input and output spaces, which leaded to an improved performance (in terms of the KRR cost (1)) over the case of using only the input space distance:

$$d([\mathbf{x}_i, \mathbf{y}_i], [\mathbf{x}_j, \mathbf{y}_j]) = \sqrt{\|\mathbf{x}_i - \mathbf{x}_j\|^2 + (\sigma_{\mathcal{X}}/\sigma_{\mathcal{Y}})\|\mathbf{y}_i - \mathbf{y}_j\|^2},$$

TABLE 1
Parameters for experiments

Mag. factor	2	3	4
M	7×7	9×9	13×13
σ_k	0.05	0.011	0.006
σ_C	0.04	0.17	0.12
λ	$0.5 \cdot 10^{-7}$	$0.1 \cdot 10^{-7}$	$0.5 \cdot 10^{-7}$
σ_N	127	80	70
σ_R	1	1	1
T_{M1}	2.2	2.2	1.1
T_{M2}	0.95	0.5	1.0

where $\sigma_{\mathcal{X}}$ and $\sigma_{\mathcal{Y}}$ are variances of distances between pairs of training data points in the input space and output space, respectively.

It should be noted that the optimization of hyper-parameters for the regression and combination parts should not be separated: choosing the hyper-parameters of regression part based on cross-validation of regression data (pairs of input and output patches) leaded to much more conservative estimation (i.e., σ_k and λ are larger) than the case of optimizing jointly the regression and combination parts. This can be explained by (further) regularization effect of the combination part which can be regarded as an instance of ensemble estimator. It has been well known that in general, ensembles of individual estimators can lead to lower variances (expectation of variance of the output for a given set of training data points) and accordingly are smoother than individual estimators (Ch. 7 of [23] and references therein). This makes the optimization criteria a non-differentiable function of hyper-parameters and prevents us from using a rather sophisticate parameter optimization methods, e.g., gradient ascent of the marginal likelihood [20].

In the experiments, we focused on the desired magnification factors at $\{2, 3, 4\}$ along each dimension. Application to other magnification factors should be straightforward. Table 1 summarizes the optimized parameters.

2.2 Post-processing Based on Image Prior

As demonstrated in Fig. 1, the result of the proposed regression-based method is significantly better than the interpolation. However, detailed visual inspection along the major edges (edges showing rapid and strong change of pixel values) reveals ringing artifacts. In general, regularization methods (depending on the specific class of regularizer) including KRR and SVR tend to fit the data with a smooth function. Accordingly, at the sharp changes of the function (edges in the case of images), either edges are smoothed or oscillation occurs to compensate the resulting loss of smoothness. This might happen for all the levels of images demonstrating the discontinuity. However, the magnitude of oscillation is in proportion to the magnitude of changes and accordingly only visible at the vicinity of major edges. While this problem can indirectly be resolved by imposing less aggressive regularization at the edges, a more direct approach is to rely on the prior knowledge of discontinuity of images. In this work, we use a modification of the natural image prior (NIP) framework proposed by Tappen et al. [1] to the pixels at the vicinity of edges:

[7] Discussion on alternative combination methods can be found in [22].

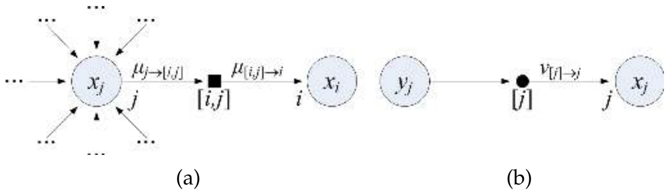


Fig. 3. Factor graph representation for the optimization of (3): (a) NIP term (message propagation from node j to node i) and (b) deviation penalty term of node j; the message from the observation variable node y_j to the factor node $[j]$ is a constant.

$$P(\{x\}|\{y\}) = \frac{1}{C}\prod_{(j,i\in\mathcal{N}_S(j))}\exp\left[-\left(\frac{|x_j-x_i|}{\sigma_N}\right)^{\alpha}\right] \cdot\prod_j\exp\left[-\left(\frac{x_j-y_j}{\sigma_R}\right)^2\right], \qquad (3)$$

where $\{y\}$ represents the observed variables corresponding to the pixel values of Y, $\{x\}$ represents the latent variable, $\mathcal{N}_S(j)$ stands for the 8-connected neighbors of the pixel location j, and C is a normalization constant. With the objective of achieving the maximum probability (equivalently, the minimum energy as the inverse of (3)) for a given image, the second product term has the role of preventing the final solution flowing far away from the input regression-based super-resolution result Y, while the first product term (NIP term) tends to smooth the image based on the costs $|\hat{x}_j - \hat{x}_i|$. The role of $\alpha(< 1)$ is to re-weight the costs such that the largest difference is stressed relatively less than the others and accordingly large changes of pixel values are relatively less penalized. Furthermore, the cost term $|\hat{x}_j-\hat{x}_i|^{\alpha}$ becomes piece-wise concave with boundary points (i.e., boundaries between concave intervals) at $\mathcal{N}_S(j)$ such that if the second term is removed, the minimum energy for a pixel j is achieved by assigning it with the value of a neighbor, rather than a certain weighted average of neighborhood values which might have been the case when $\alpha > 1$. Accordingly, this distribution prefers a strong edge rather than a set of small edges and can be used to resolve the problem of smoothing around major edges. The optimization of (3) is performed by a max-sum type belief propagation (BP) similarly to [1]. To facilitate the optimization, we reuse the candidate set generated from the regression step so that the best candidates are chosen by the BP. Accordingly, all possible outputs for each pixel location are constrained to be the N candidates generated during the regression step.

In the original NIP framework, the second term is replaced by the *reconstruction constraint* which measures the distance between the input low-resolution image and an image reconstructed from the high-resolution configuration according to the down-sampling model (blurring and sub-sampling) [1], [24]. The reconstruction constraint corresponds to a generative model, and with the suitable prior (e.g., NIP), provides a MAP framework. However, without the existence of multiple images, which might have guided better the reconstruction, relying on the reconstruction constraint in the proposed method could result in noisy images as the down-sampling process has the effect of removing noises and can make it harder to penalize the noisy configuration. (cf. [5]).[8] Furthermore, we have found that it is not straightforward to control the contribution of NIP part to prevent this effect as it often leaded to a piece-wise constant image. Accordingly, in this work, we simply penalize the deviation from the regression output (Y) which is far less noisy. The main disadvantage of the proposed scheme in comparison to the original NIP is that the intuitive probabilistic interpretation of super-resolution process [1] is no longer possible. However, on the other hand, since the resulting message structure is significantly simpler than the original version, the optimization can be made much faster:

Fig. 4. Gallery of test images (disjoint from training images): we refer to the images in the text by its position in raster order.

$$\begin{aligned}
\nu_{[j]\to j}(x_j) &= -\frac{1}{2}\left(\frac{|x_j-y_j|}{\sigma_R}\right)^2\\
\mu_{[i,j]\to i}(x_i) &= \max_{x_j}\left[\mu_{j\to[i,j]}(x_j) - \frac{1}{2}\left(\frac{|x_j-x_i|}{\sigma_N}\right)^{\alpha}\right]\\
\mu_{j\to[i,j]}(x_j) &= \nu_{[j]\to j}(x_j) + \sum_{k\in\mathcal{N}_S(j)\setminus i}\mu_{[j,k]\to j}(x_j).
\end{aligned}$$

These (logarithms of) messages can be derived from (3) based on the factor graph representation of Fig. 3. The message $\nu_{[j]\to j}$ represents the reconstruction constraint at the variable node j while the other two messages correspond to the propagation of a belief from j to i based on the NIP cost. The outgoing message $\mu_{j\to[i,j]}$ from j to the factor node $[i,j]$ is composed of the sum of $\nu_{[j]\to j}$ and all the messages from the neighboring factor nodes of j except for the node $[i,j]$. The message $\mu_{[i,j]\to i}$ is calculated as the maximum of the sum of $\mu_{j\to[i,j]}$ and (the logarithm of) the NIP cost over all the latent values x_j.

The major edges are found by thresholding each pixel based on the L^2 norm of the Laplacian and the range of pixel values in the local patches, i.e., classifying a pixel into 'major edge class' if the norm of Laplacian and the maximum difference of pixel values within a local patch are larger than thresholds T_{M1} and T_{M2}, respectively (cf. Table 1; see [22] for details of parameter optimization). While the improvements in terms of PSNR are not significant (e.g., for the case of magnification factor 2, on average 0.0003dB from the combined regression result) the improved visual quality at major edges demonstrate the effectiveness of using the prior of natural images (Figs. 1 and 5).

3 Experiments

For training and quantitative evaluation, a set of pairs of high-resolution and corresponding low-resolution images were

[8] In original work of Tappen et al. [1], the set of possible configurations is much more constrained than that of our method: candidates are 2×2-size image patches rather than individual pixels. Accordingly, in their method this problem is not as serious as the case of naïvely using the reconstruction constraint in the proposed method (cf. Appendix in the supplementary material for more discussion).

TABLE 2
Performance of different example-based super-resolution algorithms: mean improvement (standard deviation) of PSNR values from the input interpolation

Mag. factor	2	3	4
NN	0.11(0.42)	N/A	-0.85(0.56)
LLE	-0.18(0.31)	-0.17(0.45)	-0.25(0.32)
NIP	-0.50(0.51)	N/A	N/A
SVR	1.31(0.41)	0.82(0.30)	0.79(0.44)
Proposed method	1.91(0.58)	1.34(0.47)	1.15(0.56)

obtained by blurring and subsampling[9] a set of high-resolution images (the test images are shown in Fig. 4). For comparison, several different example-based image super-resolution methods were implemented, which include Freeman et al.'s fast NN-based method [7], Chang et al.'s LLE-based method [12], Tappen et al.'s NIP [1],[10] and our previous SVR-based method [15] (trained based on only 10,000 data points). Experiments with Tappen et al.'s NIP were performed only at the magnification factor 2 as it was not straightforward to implement it for the other magnification factors. For the same reason, Freeman et al.'s NN method was applied only to the case of magnification factors 2 and 4. For comparison with non-example-based methods which are not implemented by us, we performed super-resolution on several images downloaded from the websites of the authors of [3], [4], [6].[11] To obtain super-resolution results at image boundary, which are not directly available as $M > N$ for the proposed methods and similarity for other example-based methods, the input images were extended by symmetrically replicating pixel values across the image boundary. For the experiments with color images, we applied the model trained on intensity images to each RGB channel and combined them.

Figures 5 and 6 show examples of super-resolution. All the example-based super-resolution methods outperformed the spline interpolation in terms of visual plausibility. The NN-based method and the original NIP produced sharper images at the expense of introducing noise which, even with the improved visual quality, led to lower PSNR values than the interpolations (Table 2). The results of LLE are less noisy. However, it tended to smooth out texture details as observed in the third image of Fig.5(c) and accordingly produced low PSNR values. The SVR produced less noisy images, but it generated smoothed edges and perceptually distracting ring artifacts which have almost disappeared in the results of the proposed method (e.g., the first and the fourth images of Fig. 5(d)). Disregarding the post-processing stage, we measured on average 0.60dB improvement of PSNRs for the proposed method from the SVR (magnification factor 2 case). This could be attributed to the sparsity of the solution which enabled training on a large data set and the effectiveness of the candidate combination scheme. Moreover, in comparison to SVR, the proposed method requires much less processing time: super-resolving a 256×256-size image into 512×512 requires around 27 seconds for the proposed method and 18 minutes for the SVR-based method on a 3GHz machine. For quantitative comparison, PSNRs of different algorithms are summarized in Table 2.

An interesting property of NN-based method is that it introduced certain texture details which were absent in the input low-resolution images and even in the ground truth images. Sometimes, these 'pseudo textures' provided more realistic images than others (e.g., the fifth image of Fig. 5(b)). On the other hand, the proposed method did not generate such new texture details but instead provided a coherent enhancement of existing texture and edge patterns (cf. Fig. 5(g)). As noted in [4], a preference between the two techniques may depend on the specific image and subjective concerns.

In comparison with non-example-based methods of Dai et al. [3] and Fattal [4], the proposed method resulted in a better preservation of texture details and more natural transitions of pixel values across strong edges as shown in the stripe pattern of Fig. 6(c). Furthermore, the results of the proposed method look less jagged as observed in petals in the first row of Fig. 6(f).

4 Discussion

Except for the preprocessing part (interpolation and the calculation of Laplacian), the proposed method is application agnostic, i.e., the learning part is independent of specific problem at hand. In principle, this *generic* learning part can be applied to any problem when suitable examples of input and target output images are available. Accordingly, future work will include exploring the potential of learning-based approaches, including the proposed method, for various image enhancement and understanding applications. In the appendix provided in the accompanying supplementary material of the current paper, we show an application of the proposed method to artifact removal of JPEG encoded images.

Acknowledgments

The contents of this paper have been greatly benefited from discussions with Gökhan Bakır, Christian Walder, Matthew Blaschko, Christoph Lampert, Daewon Lee, and Matthias Hein and also from the comments of anonymous reviewers. The idea of using localized KRR was originated by Christian Walder. The authors would like to thank Dong Ho Kim for helping with experiments. K. I. Kim was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2005-214-D00348).

References

[1] M. F. Tappen, B. C. Russel, and W. T. Freeman, "Exploiting the sparse derivative prior for super-resolution and image demosaicing," in *Proc. IEEE Workshop on Statistical and Computational Theories of Vision*, 2003.

[2] D. Tschumperlé and R. Deriche, "Vector-valued image regularization with pdes: a common framework for different applications," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 506–517, 2005.

[3] S. Dai, M. Han, W. Xu, Y. Wu, and Y. Gong, "Soft edge smoothness prior for alpha channel super resolution," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[9] We use spline resampling which is naturally unbiased to any specific direction in the generation of low-resolution images [22].

[10] The original NIP algorithm was developed for super-resolving the pixel subsampled image. Accordingly, for the experiments with NIP, the low resolution images were generated by pixel subsampling. The visual qualities of the super-resolution results are not significantly different from the results obtained from spline resampling. However, the quantitative results should not be directly compared with other methods. The parameters used for experiments in the current work simply follow those described in [1].

[11] The original images and the results of [3], [4], and [6] are courtesy of Shengyang Dai, Raanan Fattal, and William T. Freeman, respectively.

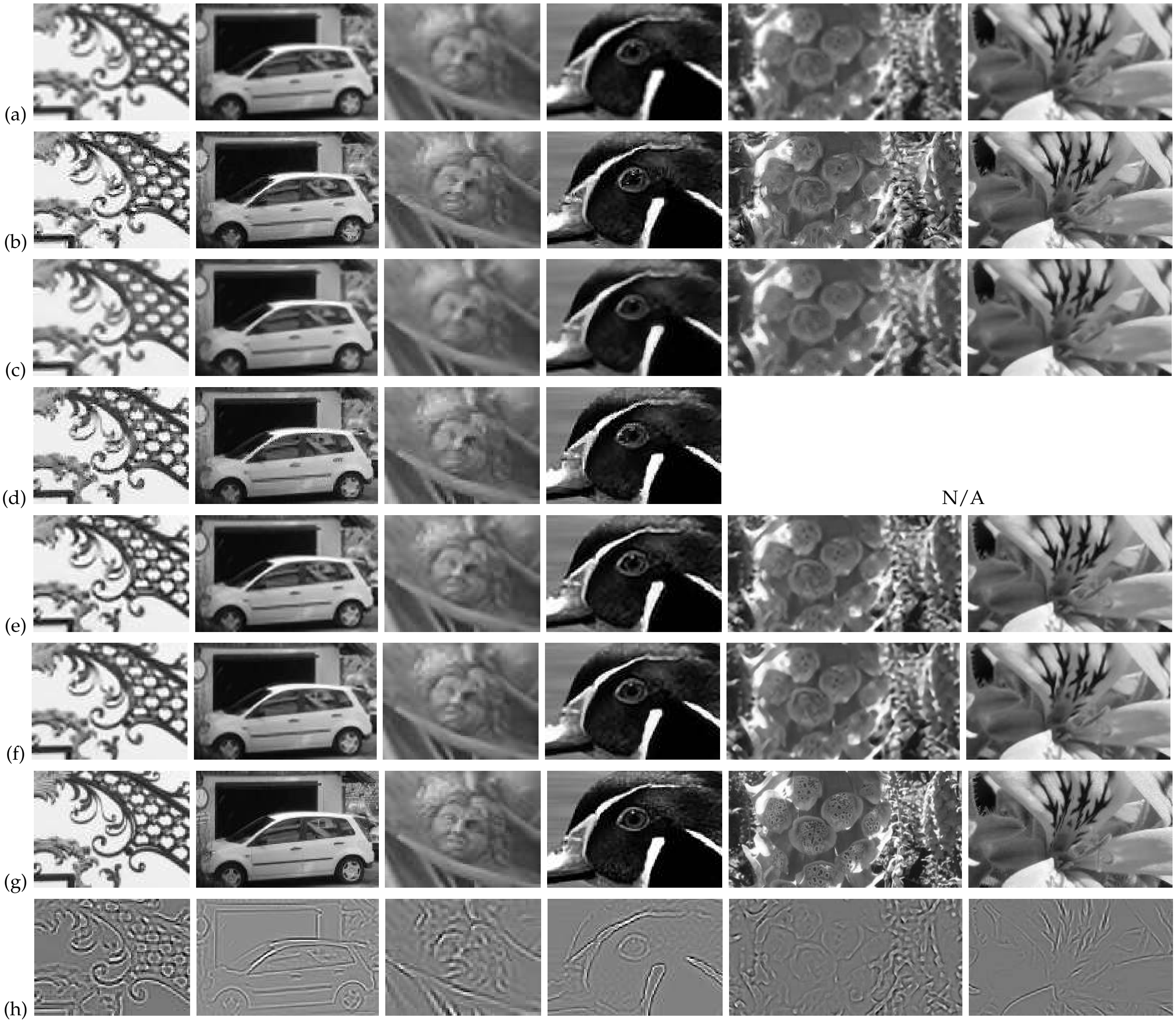


Fig. 5. Super-resolution examples of example-based algorithms: (a) interpolations, (b)-(f) super-resolution results of NN [7], LLE [12], NIP [1], SVR [15], and proposed method, respectively, (g) original high-resolution images, and (h) differences of the images in (f) and (a), respectively, which correspond to the details estimated by the proposed method. Magnification factors are 2 and 4 for the first four columns and the last two columns, respectively. Experiments with NIP [1] were performed only at the magnification factor 2 case (see texts for details). Please refer to the electronic version of the current paper for better visualization.

[4] R. Fattal, “Image upsampling via imposed edge statistics,” *ACM Trans. Graphics (Proc. SIGGRAPH 2007)*, vol. 26, no. 3, pp. 95:1–95:8, 2007.

[5] S. Baker and T. Kanade, “Limits on super-resolution and how to break them,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1167–1183, 2002.

[6] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, “Learning low-level vision,” *International Journal of Computer Vision*, vol. 40, no. 1, pp. 25–47, 2000.

[7] W. T. Freeman, T. R. Jones, and E. C. Pasztor, “Example-based super-resolution,” *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.

[8] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, “Image analogies,” in *Computer Graphics (Proc. Siggraph 2001)*. NY: ACM Press, 2001, pp. 327–340.

[9] K. I. Kim, M. O. Franz, and B. Schölkopf, “Iterative kernel principal component analysis for image modeling,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 9, pp. 1351–1366, 2005.

[10] L. C. Pickup, S. J. Roberts, and A. Zissermann, “A sampled texture prior for image super-resolution,” in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2004.

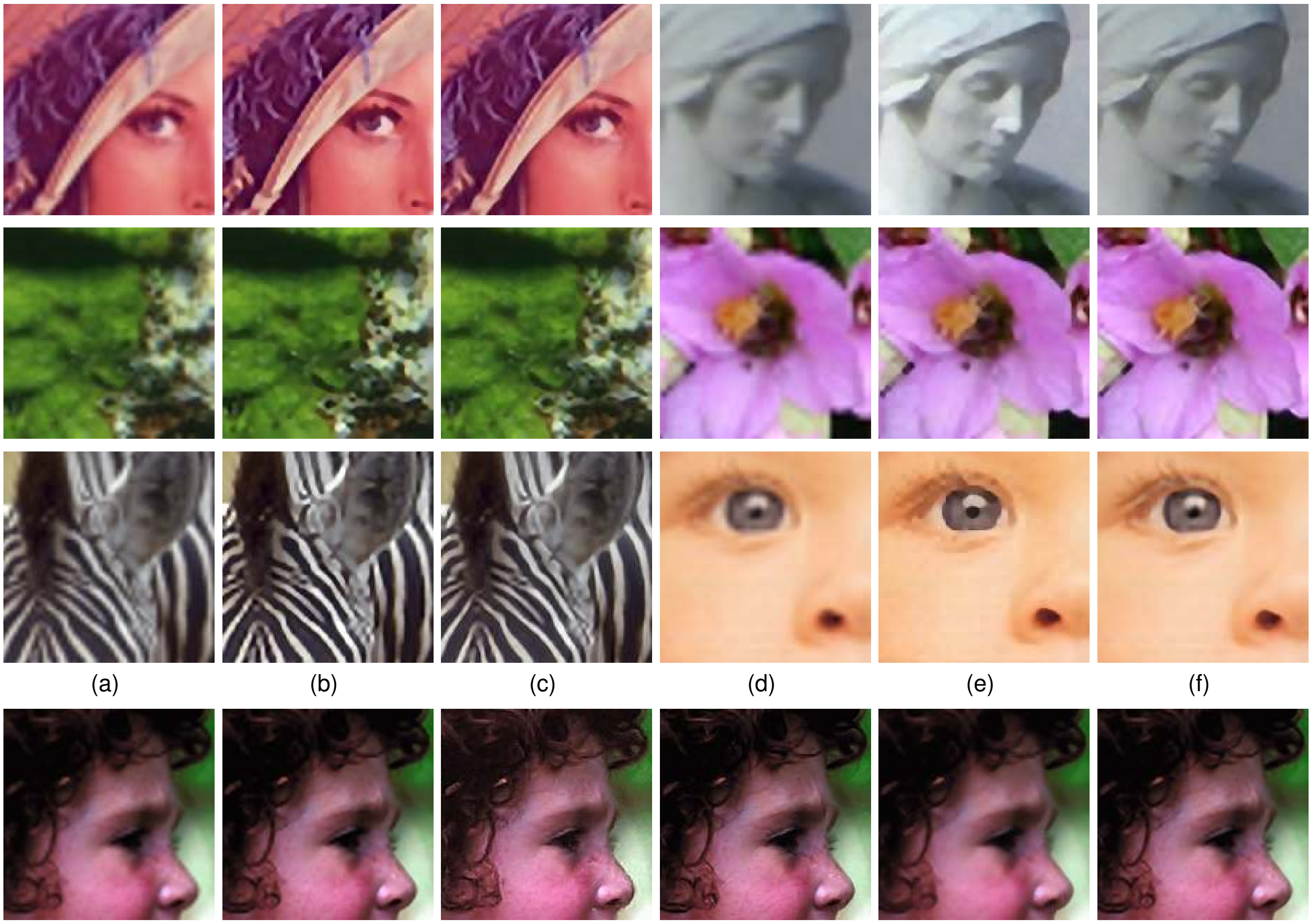
[11] C. V. Jiji and S. Chaudhuri, “Single-frame image super-resolution through contourlet learning,” *Journal of Multidimensional System and Signal Processing*, vol. 2006, pp. 1–11, 2006.

[12] H. Chang, D.-Y. Yeung, and Y. Xiong, “Super-resolution through neighbor embedding,” in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004, pp. 275–282.

[13] D. Datsenko and M. Elad, “Example-based single image super-resolution: a global MAP approach with outlier rejection,” *Journal of Multidimensional System and Signal Processing*, vol. 18, no. 2–3, pp. 103–121, 2007.

[14] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. New York: Springer-Verlag, 2001.

[15] K. I. Kim, D. H. Kim, and J.-H. Kim, “Example-based learning for image super-resolution,” in *Proc. the third Tsinghua-KAIST Joint*



(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Fig. 6. Comparison between super-resolution results of several different algorithms: (a), (d), and (g) interpolations (magnification factors 3, 4, and 4, respectively), (b) and (h) Dai et al. [3], (e) Fattal [4], (i) Freeman et al. [6], (j) Freeman et al. [7], (k) Chang et al. [12], and (c), (f), and (l) proposed method.

Workshop on Pattern Recognition, 2004, pp. 140–148.

[16] K. Ni and T. Q. Nguyen, "Image superresolution using support vector regression," *IEEE Trans. Image Processing*, vol. 16, no. 6, pp. 1596–1610, 2007.

[17] K. I. Kim and Y. Kwon, "Example-based learning for single image super-resolution," in *Proc. DAGM*, 2008, pp. 456–465.

[18] P. Vincent and Y. Bengio, "Kernel matching pursuit," *Machine Learning*, vol. 48, pp. 165–187, 2002.

[19] B. Schölkopf and A. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[20] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2006.

[21] S. S. Keerthi and W. Chu, "A matching pursuit approach to sparse gaussian process regression," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2005.

[22] K. I. Kim and Y. Kwon, "Example-based learning for single-image super-resolution and JPEG artifact removal," Max-Planck-Insitut für biologische Kybernetik, Tübingen, Tech. Rep. 173, August 2008. [Online]. Available: http://www.kyb.mpg.de/publications/attachments/TechReport-173_[0].pdf

[23] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. New Jersey: Prentice Hall, 1999.

[24] Z. Lin and H.-Y. Shum, "Fundamental limits of reconstruction-based superresolution algorithms under local translation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 83–97, 2004.