Super-Resolution Through Neighbor Embedding

Hong Chang, Dit-Yan Yeung, Yimin Xiong
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{*hongch, dyyeung, csxym@cs.ust.hk*}

Abstract

In this paper, we propose a novel method for solving single-image super-resolution problems. Given a low-resolution image as input, we recover its high-resolution counterpart using a set of training examples. While this formulation resembles other learning-based methods for super-resolution, our method has been inspired by recent manifold learning methods, particularly locally linear embedding (LLE). Specifically, small image patches in the low- and high-resolution images form manifolds with similar local geometry in two distinct feature spaces. As in LLE, local geometry is characterized by how a feature vector corresponding to a patch can be reconstructed by its neighbors in the feature space. Besides using the training image pairs to estimate the high-resolution embedding, we also enforce local compatibility and smoothness constraints between patches in the target high-resolution image through overlapping. Experiments show that our method is very flexible and gives good empirical results.

1. Introduction

1.1. Single-image super-resolution

Super-resolution is the problem of generating a high-resolution image from one or more low-resolution images. While most methods have been proposed for super-resolution based on multiple low-resolution images of the same scene (e.g., [2, 6, 8, 22]), the focus of this paper is on generating a high-resolution image from a single low-resolution image, with the help of a set of one or more training images from scenes of the same or different types. We refer to this as the *single-image super-resolution* problem.

The single-image super-resolution problem arises in a number of real-world applications. A common application occurs when we want to increase the resolution of an image while enlarging it using a digital imaging software (such as Adobe Photoshop). Another application is found in web pages with images. To shorten the response time of browsing such web pages, images are often shown in low-resolution forms (as the so-called "thumbnail images"). An enlarged, higher-resolution image is only shown if the user clicks on the corresponding thumbnail. However, this approach still requires the high-resolution image to be stored on the web server and downloaded to the user's client machine on demand. To save storage space and communication bandwidth (hence download time), it would be desirable if the low-resolution image is downloaded and then enlarged on the user's machine. Yet another application arises in the restoration of old, historic photographs, sometimes known as image inpainting [5]. Besides reverting deteriorations in the photographs, it is sometimes beneficial to also enlarge them with increased resolution for display purposes.

1.2. Related previous work

Simple resolution enhancement methods based on smoothing and interpolation techniques for noise reduction have been commonly used in image processing. Smoothing is usually achieved by applying various spatial filters such as Gaussian, Wiener, and median filters. Commonly used interpolation methods include bicubic interpolation and cubic spline interpolation [13]. Interpolation methods usually give better performance than simple smoothing methods. However, both methods are based on generic smoothness priors and hence are indiscriminate since they smooth edges as well as regions with little variations, causing blurring problems.

More recently, some learning-based methods have been proposed by different researchers. Some methods make use of a training set of images [1, 9, 10, 11, 12, 14, 19], while others do not require a training set but re-

Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)
1063-6919/04 $20.00 © 2004 **IEEE**

quire strong image priors that are either hypothesized [17] or learned from data [18].[1] The methods based on a training set are very similar in spirit. While the framework based on image analogies [12] was proposed for a wide variety of image texturing problems including super-resolution, the method is less effective than other super-resolution methods as no interaction between adjacent elements (pixels or image patches) in the high-resolution image is explicitly enforced to ensure compatibility. Nevertheless, all these methods use the training data in a similar way. In particular, each element in the target high-resolution image comes from only one nearest neighbor in the training set.

It should be noted that similar ideas have also been explored in the context of texture synthesis [7, 23, 24]. These methods typically require only one sample texture as input, possibly with some random noise [23] or a target model [24]. Similar to learning-based methods for super-resolution, only one nearest sample texture is selected to generate each synthesized pixel.

1.3. This paper

In this paper, we propose a flexible method that, in principle, can be used for super-resolution problems with arbitrary magnification factors up to some fundamental limits. More importantly, we propose a new, more general way of using the training examples, so that multiple training examples can contribute simultaneously to the generation of each image patch in the high-resolution image. This property is very important as generalization over the training examples is possible and hence fewer training examples are required.

The rest of this paper is organized as follows. In Section 2, we formulate the super-resolution problem more precisely and introduce our method based on ideas from manifold learning. Some details of the experimental setup are discussed in Section 3, including feature representation, training set, and model parameters. Experimental results are then presented in Section 4. Finally, Section 5 gives some concluding remarks.

2. Neighbor embedding

2.1. Problem formulation

The single-image super-resolution problem that we want to solve can be formulated as follows. Given a low-resolution image $\mathcal{X}_t$ as input, we estimate the target high-resolution image $\mathcal{Y}_t$ with the help of a training set of one or more low-resolution images $\mathcal{X}_s$ and the corresponding high-resolution images $\mathcal{Y}_s$.

We represent each low- or high-resolution image as a set of small overlapping image patches. $\mathcal{X}_t$ and $\mathcal{Y}_t$ have the same number of patches, and each low-resolution image in $\mathcal{X}_s$ and the corresponding high-resolution image in $\mathcal{Y}_s$ also have the same number of patches. We denote the sets of image patches corresponding to $\mathcal{X}_s$, $\mathcal{Y}_s$, $\mathcal{X}_t$ and $\mathcal{Y}_t$ as $\{\mathbf{x}_s^p\}_{p=1}^{N_s}$, $\{\mathbf{y}_s^p\}_{p=1}^{N_s}$, $\{\mathbf{x}_t^q\}_{q=1}^{N_t}$ and $\{\mathbf{y}_t^q\}_{q=1}^{N_t}$, respectively. Obviously, N_s and N_t depend on the patch size and the degree of overlap between adjacent patches.

Ideally, each patch generated for the high-resolution image $\mathcal{Y}_t$ should not only be related appropriately to the corresponding patch in the low-resolution image $\mathcal{X}_t$, but it should also preserve some inter-patch relationships with adjacent patches in $\mathcal{Y}_t$. The former determines the accuracy while the latter determines the local compatibility and smoothness of the high-resolution image. To satisfy these requirements as much as possible, we would like our method to have the following properties: (a) Each patch in $\mathcal{Y}_t$ is associated with multiple patch transformations learned from the training set. (b) Local relationships between patches in $\mathcal{X}_t$ should be preserved in $\mathcal{Y}_t$. (c) Neighboring patches in $\mathcal{Y}_t$ are constrained through overlapping to enforce local compatibility and smoothness.

2.2. Manifold learning

Our method is based on the assumption that small patches in the low- and high-resolution images form manifolds with similar local geometry in two distinct spaces. This assumption is valid because the resulting representation is stable and hence independent of the resolution as long as the embedding is isometric. Each patch, represented as a feature vector, corresponds to a point in one of the two feature spaces. For convenience, we use $\mathbf{x}_s^p$, $\mathbf{y}_s^p$, $\mathbf{x}_t^q$ and $\mathbf{y}_t^q$ to denote the feature vectors as well as the corresponding image patches, and $\mathcal{X}_s$, $\mathcal{Y}_s$, $\mathcal{X}_t$ and $\mathcal{Y}_t$ to denote the sets of feature vectors as well as the corresponding images.

Recently, some new manifold learning (or nonlinear dimensionality reduction) methods have been proposed to automatically discover low-dimensional nonlinear manifolds in high-dimensional data spaces and embed them onto low-dimensional embedding spaces, using tractable linear algebraic techniques that are not prone to local minima. These include isometric feature mapping (Isomap) [20, 21], locally linear embedding (LLE) [15, 16], and Laplacian eigenmap [3, 4]. Our super-resolution method to be described below has been

1 The model for the dynamic structure super-resolution method [18] is designed specifically for resolution doubling (i.e., 2X magnification) only.

inspired by these manifold learning methods, particularly LLE.

2.3. Locally linear embedding

LLE is a promising manifold learning method that has aroused a great deal of interest in machine learning. It computes low-dimensional, neighborhood-preserving embeddings of high-dimensional inputs and recovers the global nonlinear structure from locally linear fits.

The LLE algorithm is based on simple geometric intuitions. Suppose there are N points in a high-dimensional data space of dimensionality D, where the N points are assumed to lie on or near a nonlinear manifold of intrinsic dimensionality $d < D$ (typically $d \ll D$). Provided that sufficient data points are sampled from the manifold, each data point and its neighbors are expected to lie on or close to a locally linear patch of the manifold. The local geometry of each patch can be characterized by the reconstruction weights with which a data point is reconstructed from its neighbors.

The LLE algorithm can be summarized as follows:

1. For each data point in the D-dimensional data space:
   (a) Find the set of K nearest neighbors in the same space.
   (b) Compute the reconstruction weights of the neighbors that minimize the reconstruction error.
2. Compute the low-dimensional embedding in the d-dimensional embedding space such that it best preserves the local geometry represented by the reconstruction weights.

2.4. Our neighbor embedding method

As in LLE, local geometry is characterized in our method by how a feature vector corresponding to a patch can be reconstructed by its neighbors in the feature space. For each patch in the low-resolution image $\mathcal{X}_t$, we first compute the reconstruction weights of its neighbors in $\mathcal{X}_s$ by minimizing the local reconstruction error. The high-resolution embedding (as opposed to the low-dimensional embedding of LLE) is then estimated from the training image pairs by preserving local geometry. Finally, we enforce local compatibility and smoothness constraints between adjacent patches in the target high-resolution image through overlapping.

The neighbor embedding algorithm of our method can be summarized as follows:

1. For each patch $\mathbf{x}_t^q$ in image $\mathcal{X}_t$:
   (a) Find the set $\mathcal{N}_q$ of K nearest neighbors in $\mathcal{X}_s$.
   (b) Compute the reconstruction weights of the neighbors that minimize the error of reconstructing $\mathbf{x}_t^q$.
   (c) Compute the high-resolution embedding $\mathbf{y}_t^q$ using the appropriate high-resolution features of the K nearest neighbors and the reconstruction weights.
2. Construct the target high-resolution image $\mathcal{Y}_t$ by enforcing local compatibility and smoothness constraints between adjacent patches obtained in step 1(c).

We implement step 1(a) by using Euclidean distance to define neighborhood. Based on the K nearest neighbors identified, step 1(b) seeks to find the best reconstruction weights for each patch $\mathbf{x}_t^q$ in $\mathcal{X}_t$. Optimality is achieved by minimizing the local reconstruction error for $\mathbf{x}_t^q$

$$\mathcal{E}^q = \|\mathbf{x}_t^q - \sum_{\mathbf{x}_s^p \in \mathcal{N}_q} w_{qp}\mathbf{x}_s^p\|^2, \tag{1}$$

which is the squared distance between $\mathbf{x}_t^q$ and its reconstruction, subject to the constraints $\sum_{\mathbf{x}_s^p \in \mathcal{N}_q} w_{qp} = 1$ and $w_{qp} = 0$ for any $\mathbf{x}_s^p \notin \mathcal{N}_q$. Apparently, minimizing $\mathcal{E}^q$ subject to the constraints is a constrained least squares problem. Let us define a local Gram matrix $\mathbf{G}_q$ for $\mathbf{x}_t^q$ as

$$\mathbf{G}_q = (\mathbf{x}_t^q\mathbf{1}^T - \mathbf{X})^T(\mathbf{x}_t^q\mathbf{1}^T - \mathbf{X}),$$

where $\mathbf{1}$ is a column vector of ones and $\mathbf{X}$ is a $D \times K$ matrix with its columns being the neighbors of $\mathbf{x}_t^q$. Moreover, we group the weights of the neighbors to form a K-dimensional weight vector $\mathbf{w}_q$ by reordering the subscript p of each weight w_{qp}. The constrained least squares problem has the following closed-form solution:

$$\mathbf{w}_q = \frac{\mathbf{G}_q^{-1}\mathbf{1}}{\mathbf{1}^T\mathbf{G}_q^{-1}\mathbf{1}}.$$

Instead of inverting $\mathbf{G}_q$, a more efficient way is to solve the linear system of equations $\mathbf{G}_q\mathbf{w}_q = \mathbf{1}$, and then normalize the weights so that $\sum_{\mathbf{x}_s^p \in \mathcal{N}_q} w_{qp} = 1$. After repeating steps 1(a) and 1(b) for all N_t patches in $\mathcal{X}_t$, the reconstruction weights obtained form a weight matrix $\mathbf{W} = [w_{qp}]_{N_t \times N_s}$.

Step 1(c) computes the initial value of $\mathbf{y}_t^q$ based on $\mathbf{W}$:

$$\mathbf{y}_t^q = \sum_{\mathbf{x}_s^p \in \mathcal{N}_q} w_{qp}\mathbf{y}_s^p. \tag{2}$$

In step 2, we use a simple method to enforce inter-patch relationships by averaging the feature values in overlapped regions between adjacent patches. Other more sophisticated methods may also be used.

3. Experiments

3.1. Feature representation

As discussed above, each image patch is represented by a feature vector. In this subsection, we will address the issue of feature representation for both low- and high-resolution images.

Color images are commonly represented by the RGB channels. However, humans are more sensitive to changes in luminance than to changes in color. Thus, instead of using the RGB color model, we use the YIQ color model where the Y channel represents luminance and the I and Q channels represent chromaticity. Conversion between the RGB and YIQ color schemes can be done easily via a linear transformation. In our method, the chromaticity components from the I and Q channels are not learned. They are simply copied from the low-resolution image to the target high-resolution image. Hence, only the luminance values from the Y channel are used to define features.

For the low-resolution images, one possible scheme is to define the feature vector as a concatenation of the luminance values of all pixels inside the corresponding patch. However, this simple scheme is not satisfactory. An alternative scheme, which we use here, is to consider the relative luminance changes within a patch. This feature representation scheme allows us to use a relatively small training set. More specifically, we use the first-order and second-order gradients of the luminance as features. Figure 1 shows a 5×5 local neighborhood of the pixel at the center with luminance value z_{13}. The first-order gradient vector of z_{13}, denoted ∇z_{13}, and the second-order gradient vector, denoted $\nabla^2 z_{13}$, can easily be derived as follows:

$$\begin{aligned} \nabla z_{13} &= \begin{bmatrix} (z_{14} - z_{13}) + (z_{13} - z_{12}) \\ (z_{18} - z_{13}) + (z_{13} - z_8) \end{bmatrix} \\ &= \begin{bmatrix} z_{14} - z_{12} \\ z_{18} - z_8 \end{bmatrix}, \end{aligned}$$

$$\begin{aligned} \nabla^2 z_{13} &= \begin{bmatrix} (z_{15} - z_{13}) - (z_{13} - z_{11}) \\ (z_{23} - z_{13}) - (z_{13} - z_3) \end{bmatrix} \\ &= \begin{bmatrix} z_{15} - 2z_{13} + z_{11} \\ z_{23} - 2z_{13} + z_3 \end{bmatrix}. \end{aligned}$$

By combining the two gradient vectors above, we obtain four features for each pixel.[2] The feature vector for each patch is then defined as the simple concatenation of the features for all pixels within the patch. For an $n \times n$ low-resolution patch, its feature vector has $4n^2$ features.

z_1	z_2	z_3	z_4	z_5
z_6	z_7	z_8	z_9	z_{10}
z_{11}	z_{12}	z_{13}	z_{14}	z_{15}
z_{16}	z_{17}	z_{18}	z_{19}	z_{20}
z_{21}	z_{22}	z_{23}	z_{24}	z_{25}

Figure 1. A 5×5 local neighborhood in the low-resolution image for computing the first-order and second-order gradients of the pixel at the center with luminance value z_{13}.

For the high-resolution images, we define the features for each patch based only on the luminance values of the pixels in the patch. Since the features used for the low-resolution patches cannot reflect the absolute luminance, we subtract the mean value from the luminance-based feature vector of each high-resolution patch. When we construct the target high-resolution image, the mean luminance value of the corresponding low-resolution patch will be added.

3.2. Training set and model parameters

In our experiments, we use only very small training sets. There are two settings that we have explored. The first setting uses a separate set of training images. Figure 2 shows the images used in some of the experiments. These images are from the Kodak web site.[3] We have also investigated another setting where a small portion of the target high-resolution image is known and is available as the (only) training image. Given the high-resolution training image(s) $\mathcal{X}_s$, we obtain the corresponding low-resolution image(s) $\mathcal{Y}_s$ through blurring and then downsampling. Under the second setting, since the training set is very small, each patch is represented as eight different feature vectors through rotation to different orientations ($0°$, $90°$, $180°$ and $270°$) and also obtaining their mirror images. This scheme could be applied to the first setting as well, but we have not done this in our experiments.

Our method has only three parameters to determine. The first parameter is the number of nearest neighbors K for neighbor embedding. Our experiments show that the super-resolution result is not very sensi-

2 Feature weighting may be applied through weighted combination of the two feature types. However, this requires introducing an additional parameter and solving the associated model selection problem by determining the best parameter value.

3 http://www.kodak.com/go/photoquilt

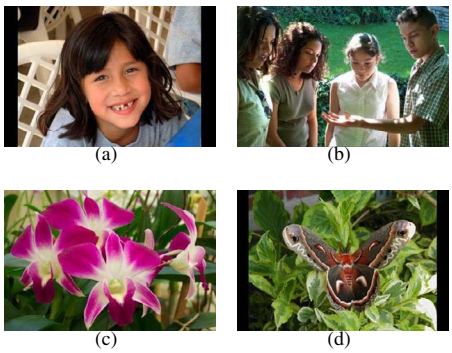

Figure 2. Training images used in some of the experiments.

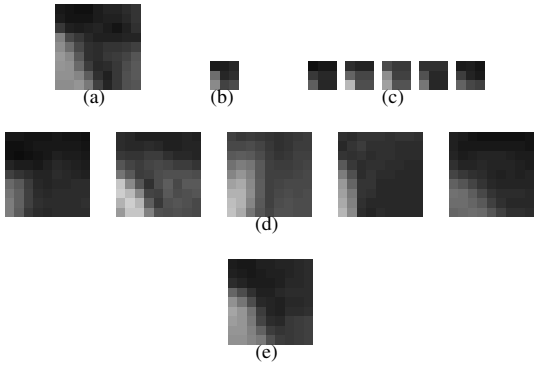

Figure 3. Neighbor embedding procedure applied to a low-resolution patch for 3X magnification: (a) true high-resolution patch; (b) input low-resolution patch downsampled from (a); (c) five nearest-neighbor low-resolution patches from the training images; (d) high-resolution patches from the training images corresponding to the low-resolution patches in (c); (e) target high-resolution patch constructed from (d).

tive to the choice of K. We set K to 5 for all our experiments. The second and third parameters are the patch size and the degree of overlap between adjacent patches. For the low-resolution images, we use 3×3 patches with an overlap of one or two pixels between adjacent patches. If we want to magnify a low-resolution image by N times in each dimension, then we use $3N \times 3N$ patches in the high-resolution image with an overlap of N or $2N$ pixels between adjacent patches.

3.3. An illustrative example

For illustration, Figure 3 shows the results of applying neighbor embedding to a small 3×3 patch from a low-resolution plant image (see Figure 7). The input low-resolution patch in (b) is downsampled from a blurred version of the true high-resolution patch in (a). Using the feature representation described above[4], five nearest-neighbor patches in (c) are obtained from the training images and their reconstruction weights are computed according to Equation (1). Based on the five corresponding high-resolution patches as shown in (d), the target high-resolution patch in (e) is constructed according to Equation (2). The reconstructed high-resolution patch is perceptually very similar to the true high-resolution patch. None of the nearest-neighbor high-resolution patches is better than this reconstructed patch, showing the potential of our method in generalizing over the training images. This provides a sound justification for the satisfactory performance even when a small training set is used.

4 Recall that different features are used for the low- and high-resolution patches.

4. Experimental results

We compare our method with median filtering and cubic spline interpolation (or bicubic interpolation in one case) for different super-resolution examples. For some examples which have been studied by other researchers, we also include their results here for comparison.

Figure 4 shows the results of applying different super-resolution methods to a head image to obtain 4X magnification. All images except those of our method are from the researchers' web site.[5] Freeman *et al.*'s method [9] and our method give the best results. While Freeman *et al.*'s method makes the reconstructed high-resolution image somewhat "non-photorealistic" near the hair region, our method smooths the texture on the face.

Figure 5 shows a 2X magnification example for a lizard image. We include results using median filtering and bicubic interpolation for comparison. We also include the result of Storkey's dynamic structure super-resolution method [18]. All images except that of our method are from his web site.[6] For our method, we

5 http://www.ai.mit.edu/people/wtf/superres

6 http://www.anc.ed.ac.uk/~amos/superresolution.html

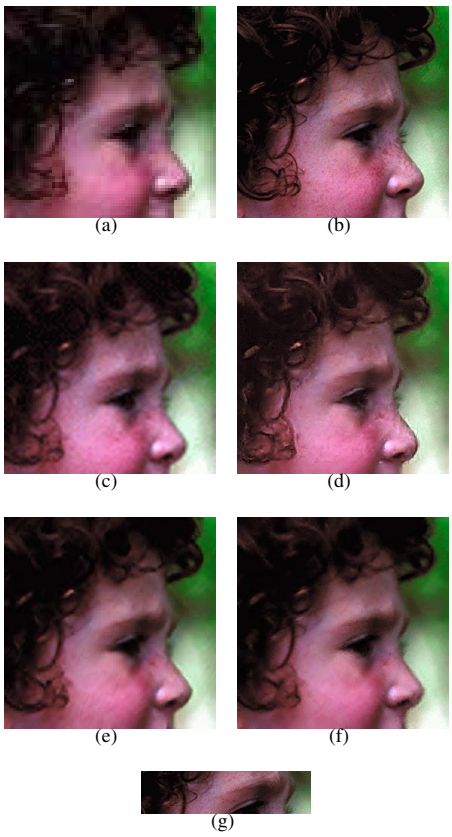

Figure 4. 4X magnification of the head image from a 70×70 low-resolution image: (a) input low-resolution image; (b) true high-resolution image; (c) cubic spline interpolation, (d) Freeman *et al.*'s method; (e) our method with training examples shown in Figures 2(a) and (b); (f) our method with part of the true high-resolution image as training example; (g) training image for (f).

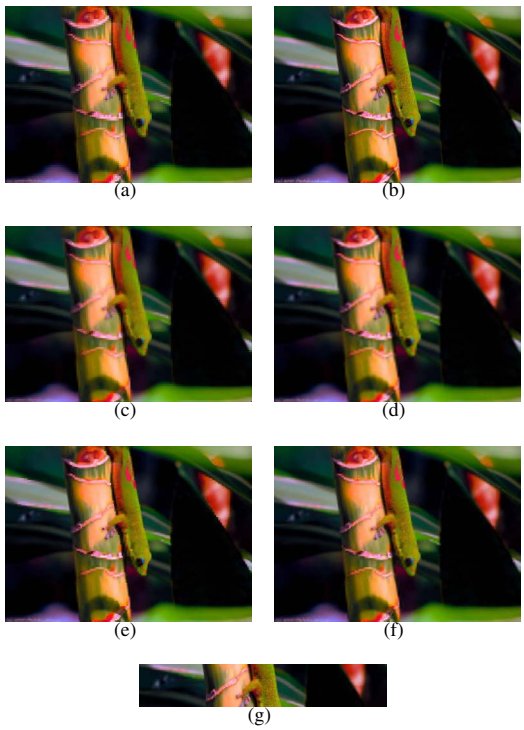

Figure 5. 2X magnification of the lizard image: (a) input low-resolution image; (b) true high-resolution image; (c) median filtering; (d) bicubic interpolation; (e) Storkey's method; (f) our method with part of the true high-resolution image as training example; (g) training image for (f).

crop a small part of the true high-resolution image as training image. As we can see, our method is clearly superior to median filtering and bicubic interpolation. Storkey's method and our method give comparable results, although our method shows better effect on some edges.

More examples using a separate set of training images are shown in Figure 6 and Figure 7. These examples are from the same Kodak web site as those in Figure 2. Both examples perform 3X magnification and use the two images in Figures 2(c) and (d) as training examples. It is clear that our method outperforms median filtering and cubic spline interpolation for both examples.

To see how the performance changes as a function of the number of nearest neighbors K used, we measure the RMS error between the super-resolution image generated and the ground-truth image as K varies over a range. Figure 8 shows the results for four examples discussed above. As we can see, the RMS error attains its lowest value when K is 4 or 5, showing that using multiple nearest neighbors (as opposed to only one nearest neighbor as in the existing methods) does give improved results.

5. Conclusion

In this paper, we have proposed a novel method for single-image super-resolution problems. While our

Figure 6. 3X magnification of the bird image: (a) input low-resolution image; (b) true high-resolution image; (c) median filtering; (d) cubic spline interpolation; (e) our method with training images shown in Figures 2(c) and (d).

Figure 7. 3X magnification of the plant image: (a) input low-resolution image; (b) true high-resolution image; (c) median filtering; (d) cubic spline interpolation; (e) our method with training images shown in Figures 2(c) and (d).

method resembles other learning-based methods in relying on a training set, our method is novel in that it uses the training images in a more general way. More specifically, generation of a high-resolution image patch does not depend on only one of the nearest neighbors in the training set. Instead, it depends simultaneously on multiple nearest neighbors in a way similar to LLE for manifold learning. An important implication of this property is that generalization over the training examples is possible and hence we can expect our method to require fewer training examples than other learning-based super-resolution methods.

Besides, we believe the use of first-order and second-order gradients of the luminance as features can better preserve high-contrast intensity changes while trying to satisfy the smoothness constraints. We may even go further by extending our method with the use of primal sketch priors, an interesting and useful idea recently proposed by Sun *et al.* [19]. We believe this extension not only can handle image primitives (e.g., edges) better, but it can also lead to significant speedup as only regions with primitives have to be transformed. We will pursue this interesting direction in our future research.

References

[1] C.B. Atkins, C.A. Bouman, and J.P. Allebach. Tree-based resolution synthesis. In *Proceedings of the Conference on Image Processing, Image Quality and Image Capture Systems*, pages 405–410, Savannah, GA, USA, 25–28 April 1999.

[2] S. Baker and T. Kanade. Limits on super-resolution and how to break them. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 372–379, Hilton Head Island, SC, USA, 13–15 June 2000.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, Cambridge, MA, USA, 2002.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[5] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the Twenty-Seventh Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2000)*, pages 417–424, New Orleans, LA, USA, 23–28 July 2000.

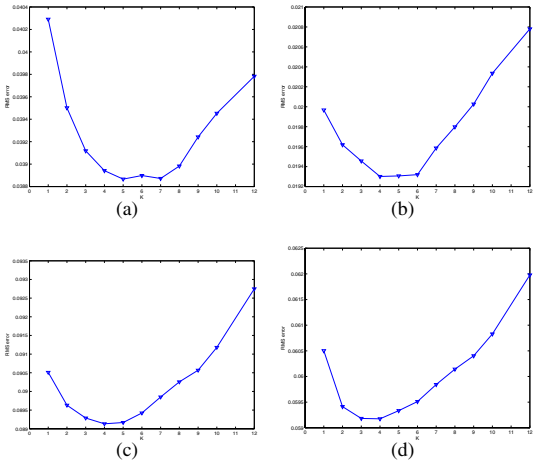

Figure 8. RMS error between the super-resolution image generated and the ground-truth image as a function of the number of nearest neighbors used: (a) head image; (b) lizard image; (c) bird image; (d) plant image.

[6] P. Cheeseman, B. Kanefsky, R. Kraft, J. Stutz, and R. Hanson. Super-resolved surface reconstruction from multiple images. Technical Report FIA-94-12, Artificial Intelligence Research Branch, NASA Ames Research Center, Moffett Field, CA, USA, December 1994.

[7] A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1033–1038, Kerkyra, Greece, 20–27 September 1999.

[8] M. Elad and A. Feuer. Super-resolution reconstruction of image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):817–834, 1999.

[9] W.T. Freeman and E.C. Pasztor amd O.T. Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47, 2000.

[10] W.T. Freeman, T.R. Jones, and E.C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002.

[11] W.T. Freeman and E.C. Pasztor. Learning low-level vision. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1182–1189, Kerkyra, Greece, 20–27 September 1999.

[12] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, and D.H. Salesin. Image analogies. In *Proceedings of the Twenty-Eighth Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2001)*, pages 327–340, Los Angeles, CA, USA, 12–17 August 2001.

[13] R.G. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1153–1160, 1981.

[14] C. Liu, H.Y. Shum, and C.S. Zhang. A two-step approach to hallucinating faces: global parametric model and local nonparametric model. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 192–198, Kauai, HI, USA, 8–14 December 2001.

[15] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[16] L.K. Saul and S.T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[17] R.R. Schultz and R.L. Stevenson. A Bayesian approach to image expansion for improved definition. *IEEE Transactions on Image Processing*, 3(3):233–242, 1994.

[18] A.J. Storkey. Dynamic structure super-resolution. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1295–1302. MIT Press, Cambridge, MA, USA, 2003.

[19] J. Sun, N.N. Zheng, H. Tao, and H.Y. Shum. Image hallucination with primal sketch priors. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 729–736, Madison, WI, USA, 18–20 June 2003.

[20] J.B. Tenenbaum. Mapping a manifold of perceptual observations. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 682–688. MIT Press, 1998.

[21] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[22] M.E. Tipping and C.M. Bishop. Bayesian image super-resolution. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1279–1286. MIT Press, Cambridge, MA, USA, 2003.

[23] L.Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the Twenty-Seventh Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2000)*, pages 479–488, New Orleans, LA, USA, 23–28 July 2000.

[24] L.Y. Wei and M. Levoy. Texture synthesis over arbitrary manifold surfaces. In *Proceedings of the Twenty-Eighth Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2001)*, pages 355–360, Los Angeles, CA, USA, 12–17 August 2001.