Example-Based Super-Resolution

William T. Freeman, Thouis R. Jones, and Egon C. Pasztor
Mitsubishi Electric Research Labs

To address the lack of resolution independence in most models, we developed a fast and simple one-pass, training-based super-resolution algorithm for creating plausible high-frequency details in zoomed images.

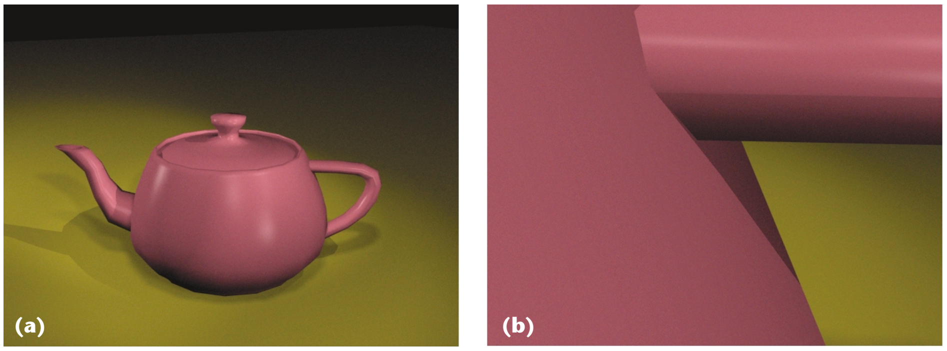
Polygon-based representations of 3D objects offer resolution independence over a wide range of scales. With this approach, object boundaries remain sharp when we zoom in on an object until very close range, where faceting appears due to finite polygon size (see Figure 1). However, constructing polygon models for complex, real-world objects can be difficult. Image-based rendering (IBR), a complementary approach for representing and rendering objects, uses cameras to obtain rich models directly from real-world data. Unfortunately, these representations no longer have resolution independence. When we enlarge a bitmapped image, we get a blurry result. Figure 2 shows the problem for an IBR version of a teapot image, rich with real-world detail. Standard pixel interpolation methods, such as pixel replication (Figures 2b and 2c) and cubic-spline interpolation (Figures 2d and 2e), introduce artifacts or blur edges. For images enlarged three octaves (factors of two) such as these, sharpening the interpolated result has little useful effect (Figures 2f and 2g).

We call methods for achieving high-resolution enlargements of pixel-based images *super-resolution* algorithms. Many applications in graphics or image processing could benefit from such resolution independence, including IBR, texture mapping, enlarging consumer photographs, and converting NTSC video content to high-definition television. We built on another training-based super-resolution algorithm[1] and developed a faster and simpler algorithm for one-pass super-resolution. (The one-pass, example-based algorithm gives the enlargements in Figures 2h and 2i.) Our algorithm requires only a nearest-neighbor search in the training set for a vector derived from each patch of local image data. This one-pass super-resolution algorithm is a step toward achieving resolution independence in image-based representations. We don't expect perfect resolution independence—even the polygon representation doesn't have that—but increasing the resolution independence of pixel-based representations is an important task for IBR.

Example-based approaches

Super-resolution relates to image interpolation—how should we interpolate between the digital samples of a photograph? Researchers have long studied this problem, although only recently with machine learning or sampling approaches. (See the "Related Approaches" sidebar for more details.)

Three complimentary ways exist for increasing an image's apparent resolution:



1 (a) When we model an object with traditional polygon techniques, it lacks some of the richness of real-world objects but behaves properly under enlargement. (b) The teapot's edge remains sharp when we enlarge it.

Related Approaches

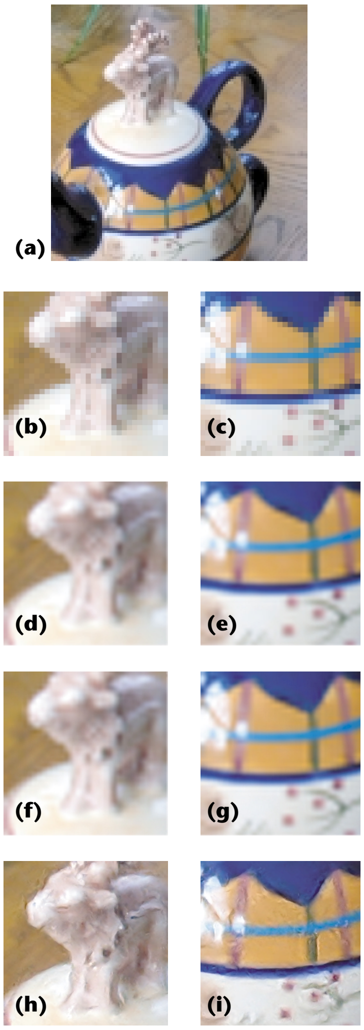
The cubic spline[1] is a common image interpolation function, but it suffers from blurring edges and image details. Recent attempts to improve on cubic-spline interpolation[2-4] have met with limited success. Schreiber[3] proposed a sharpened Gaussian interpolator function to minimize information spillover between pixels and optimize flatness in smooth areas. Schultz and Stevenson[5] used a Bayesian method for super-resolution, but it hypothesizes rather than learns the prior probability.

These analytic approaches often suffer from perceived loss of detail in textured regions. A proprietary, undisclosed algorithm, Altamira Genuine Fractals 2.0 (an Adobe Photoshop plug-in, http://www.altamira.com), does as well as any of the nontraining-based methods, but can cause blurring in texture regions and at fine lines. Recently, image interpolation-based level-set methods[6] have shown excellent results for edges.

References

1. R. Keys, "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. 29, no. 6, 1981, pp. 1153-1160.
2. F. Fekri, R.M. Mersereau, and R.W. Schafer, "A Generalized Interpolative Vq Method for Jointly Optimal Quantization and Interpolation of Images," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing* (ICASSP), vol. 5, IEEE Press, Piscataway, N.J., 1998, pp. 2657-2660.
3. W.F. Schreiber, *Fundamentals of Electronic Imaging Systems*, Springer-Verlag, New York, 1986.
4. S. Thurnhofer and S. Mitra, "Edge-Enhanced Image Zooming," *Optical Engineering*, vol. 35, no. 7, July 1996, pp. 1862-1870.
5. R.R. Schultz and R.L. Stevenson, "A Bayesian Approach to Image Expansion for Improved Definition," *IEEE Trans. Image Processing*, vol. 3, no. 3, May 1994, pp. 233-242.
6. B. Morse and D. Schwartzwald, "Image Magnification Using Levelset Reconstruction," *Proc. International Conf. Computer Vision* (ICCV), IEEE CS Press, Los Alamitos, Calif., 2001, pp. 333-341.

- Sharpening by amplifying existing image details. This is the change in the spatial frequency amplitude spectrum of an image associated with image sharpening. Existing high frequencies in the image are amplified. This is often useful to do, provided noise isn't amplified.
- Aggregating from multiple frames. Extracting a single high-resolution frame from a sequence of low-resolution video images adds value and is sometimes referred to as super-resolution.



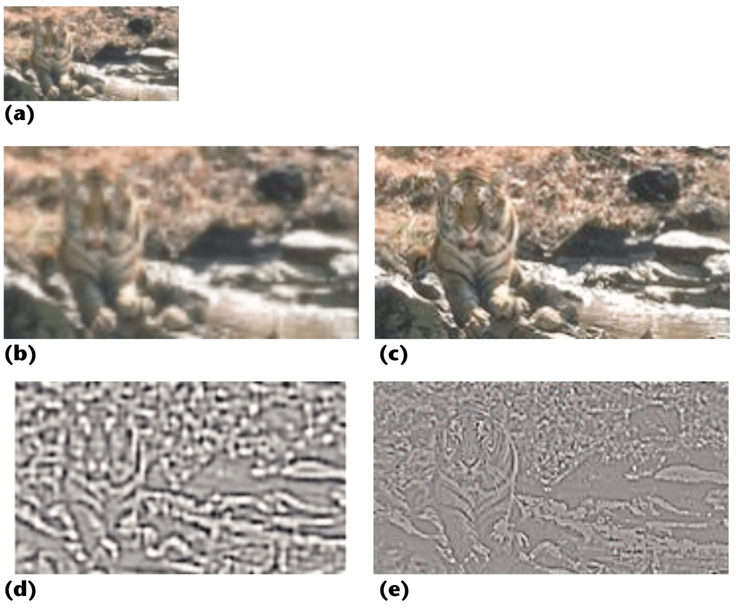
2 **(a) An image (100 × 100 pixels) of a real-world teapot shows a richness of texture but yields a blocky or blurred image when we enlarge it by a factor of 8 in each dimension by (b, c) pixel replication or (d, e) cubic-spline interpolation. ((b) through (i) were 32 × 32 pixel original subimages, zoomed by 8 to 256 × 256 images). Sharpening the cubic-spline interpolation might not help to increase the perceptual sharpness; we used the "sharpen more" option in Adobe Photoshop (f, g). (h, i) The results of our one-pass super-resolution algorithm, maintaining edge and line sharpness as well as inventing plausible texture details.**

- Single-frame super-resolution. The goal of this article is to estimate missing high-resolution detail that isn't present in the original image, and which we can't make visible by simple sharpening.

We feel researchers should use each method wherever applicable, but in this article, we focus on single-frame super-resolution. Although integrating resolution information over multiple frames is sometimes called super-resolution, for the purposes of this article, super-resolution will refer to the single-frame enlargement problem.

Because the richness of real-world images is difficult to capture analytically, for the past several years, we've been exploring a learning-based approach for enlarging images.[1-3] In a training set, the algorithm learns the fine details that correspond to different image regions seen at a low-resolution and then uses those learned relationships to predict fine details in other images. (Recently, Hertzmann et. al[4] also used a training-based method to perform super-resolution, in the context of analogies between images. Baker and Kanade[5] focused on enlarging images of a known model class—for example, faces. Liu et. al[6] built on their and our work.)

To understand why this approach should work at all,



3 Image preprocessing steps for training images. (a) We start from a low-resolution image and (c) its corresponding high-resolution source. (b) We form an initial interpolation of the low-resolution image to the higher pixel sampling resolution. In the training set, we store corresponding pairs of patches from (d) and (e), which are the band-pass or high-pass filtered and contrast normalized versions of (b) and (c), respectively. This processing allows the same training examples to apply in different image contrasts and low-frequency offsets.

consider that a collection of image pixels are special signals that have much less variability than a corresponding set of completely random variables. Researchers have studied these regularities to account for the early processing stages of the mammalian visual systems.[7,8] We exploit these regularities in our algorithms as well. We use small pieces of training images, modified for generalization by appropriate preprocessing, to create plausible image information in other images. Without restriction to a specific class of training images, it's unreasonable to expect to generate the correct high-resolution information. We aim for the more attainable goal of synthesizing visually plausible image details, such as sharp edges, and plausible looking texture.

Training set generation

To generate our training set, we start from a collection of high-resolution images and degrade each of them in a manner corresponding to the degradation we plan to undo in the images we later process. Typically, we blur and subsample them to create a low-resolution image of one-half the number of original pixels in each dimension (one-quarter the total number of pixels). To change resolution by higher factors, we typically use the single-octave algorithm recursively.

We apply an initial analytic interpolation, such as cubic spline, to the low-resolution image. This generates an image of the desired number of pixels that lacks high-resolution detail. In our training set, we only need to store the differences between the image's cubic-spline interpolation and the original high-resolution image. Figures 3a and 3c show low- and high-resolution versions of an image. Figure 3b is the initial interpolation (bilinear for this example).

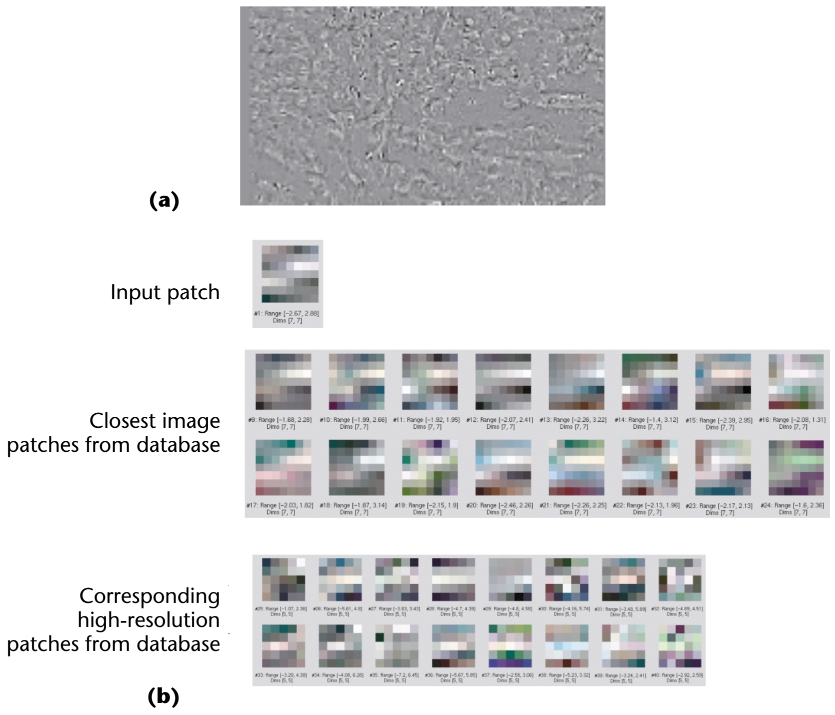
We want to store the high-resolution patch corresponding to every possible low-resolution image patch; these patches are typically 5×5 and 7×7 pixels, respectively. Even restricting ourselves to plausible image information, this is a huge amount of information to store, so we must preprocess the images to remove variability and make the training sets as generally applicable as possible.

We believe that the highest spatial-frequency components of the low-resolution image (Figure 3b) are most important in predicting the extra details in Figure 3c. We filter out the lowest frequency components in Figure 3b so that we don't have to store example patches for all possible lowest frequency component values. We also believe that the relationship between high- and low-resolution image patches is essentially independent of local image contrast. We don't want to have to store examples of that underlying relationship for all possible values of the local image contrast. Therefore, we apply a local contrast normalization, which we describe later on in the "Prediction" section. In Figures 3d and 3e, we used the resulting band-pass filtered and contrast normalized image pairs for training. We undo the contrast normalization step when we reconstruct the high-resolution image.

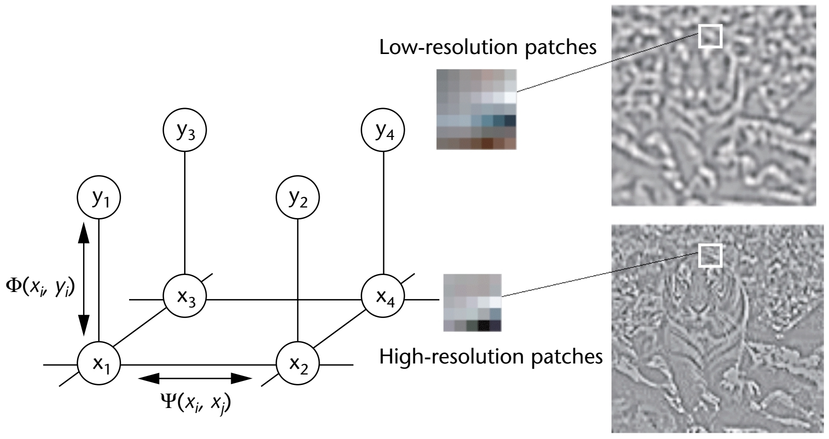
Super-resolution algorithms

If local image information alone were sufficient to predict the missing high-resolution details, we would be able to use the training set patches by themselves for super-resolution. For a given input image we want to enlarge, we would apply the preprocessing steps, break the image into patches, and look-up the missing high-resolution detail. Unfortunately, that approach doesn't work, as Figure 4a illustrates. The resulting high-resolution detail image looks like oatmeal. The local patch alone isn't sufficient to estimate plausible looking high-resolution detail.

Figure 4b illustrates why the local method doesn't work. For a given low-resolution input patch, we searched a typical training database of approximately 100,000 patches to find the 16 closest examples to the input patch (see the second line in Figure 4b). Each of these looks fairly similar to the input patch. The bottom row shows the high-resolution detail corresponding to each of these training examples; each of those looks fairly different from the other. This illustrates that local


(a)
Input patch
Closest image patches from database
Corresponding high-resolution patches from database
(b)

4 (a) Estimated high frequencies for the tiger image (Figure 3e shows the true high frequencies) formed by substituting the high frequencies of the closest training patch to Figure 3d. The lack of a recognizable image indicates that an algorithm using only local low-resolution information is insufficient; we must also use spatial context. (b) An input patch and similar low-resolution (middle rows) and paired high-resolution (bottom rows) patches. For many of these similar low-resolution patches, the high-resolution patches are different, reinforcing the lesson from (a).



5 Markov network model for the super-resolution problem. The low-resolution patches at each node y_i are the observed input. The high-resolution patch at each node x_i is the quantity we want to estimate.

patch information alone is insufficient for super-resolution, and we must take into account spatial neighborhood effects.

We explored two different approaches to exploit neighborhood relationships in super-resolution algorithms. The first uses a Markov network to probabilistically model the relationships between high- and low-resolution patches, and between neighboring high-resolution patches.[1-3] It uses an iterative algorithm, which usually converges quickly. The second approach, which we describe in detail in this article, is a one-pass algorithm that uses the same local relationship information as the Markov network. It's a fast, approximate solution to the Markov network.

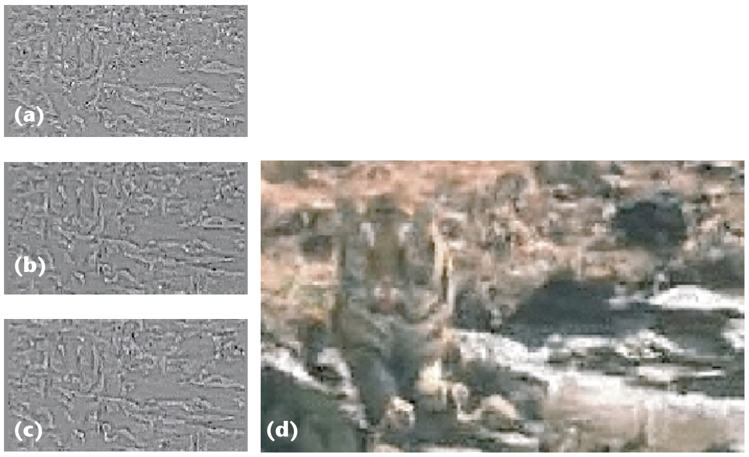
Markov network

We model the spatial relationships between patches using a Markov network, which has many well-known uses in image processing.[9] In Figure 5, the circles represent network nodes, and the lines indicate statistical dependencies between nodes. We let the low-resolution image patches be observation nodes, y. We select the 16 or so closest examples to each input patch as the different states of the hidden nodes, x, that we seek to estimate. For this network, the probability of any given high-resolution patch choice for each node is proportional to the product of all sets of compatibility matrices ψ relating the possible states of each pair of neighboring hidden nodes, and vectors ϕ relating each observation to the underlying hidden states:

$$P(x \mid y) = \frac{1}{Z} \prod_{(ij)} \psi_{ij}(x_i, x_j) \prod_i \phi_i(x_i, y_i) \qquad (1)$$

Z is a normalization constant, and the first product is over all neighboring pairs of nodes, i and j. y_i and x_i are the observed low-resolution and estimated high-resolution patches at node i, respectively.

To specify the Markov network's $\psi_{ij}(x_i, x_j)$ functions,



6 Belief-propagation solution to the Markov network for super-resolution. (a, b, and c) Estimated high frequencies after 0, 1, and 3 iterations of belief propagation. (d) Estimated full-resolution image. We applied the inverse of the contrast normalization we used in Figure 3d to (c). We added the result to Figure 3b to obtain (d). The training set for this image was two categories of the Corel database, including other tigers, but not this image.[1]

we use a simple trick.[1] We sample the input image's nodes so that the high-resolution patches overlap with each other by one or more pixels. In the overlap region, the pixel values of compatible neighboring patches should agree. We measure $d_{ij}(x_i, x_j)$, the sum of squared differences between patch candidates x_i and x_j in their overlap regions at nodes i and j. The compatibility matrix between nodes i and j is then

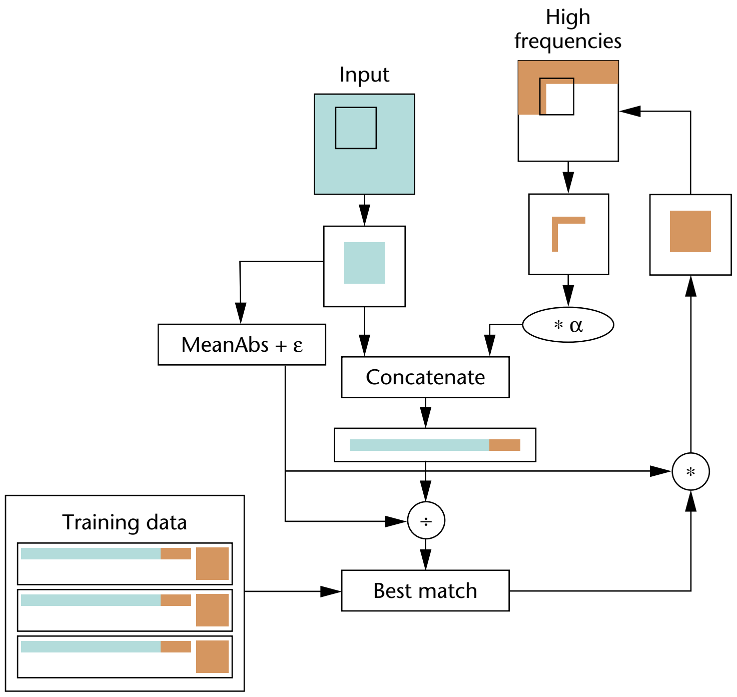
$$\psi_{ij}\left(x_i, x_j\right) = \exp\left(-\frac{d_{ij}\left(x_i, x_j\right)}{2\sigma^2}\right)$$

where σ is a noise parameter. We use a similar quadratic penalty on differences between the observed low-resolution image patch, y_i, and the candidate low-resolution patch found from the training set, x_i, to specify the Markov network compatibility function, $\phi_i(x_i, y_i)$.

The optimal high-resolution patches at each node is the collection that maximizes the Markov network's probability. Finding the exact solution can be computationally intractable, but we've found good results using the approximate solution obtained by running a fast, iterative algorithm called *belief propagation*. Typically, three or four iterations of the algorithm are sufficient (see Figure 6).

The belief-propagation algorithm updates "messages," $\mathbf{m}_{ij}$ from node i to node j, which are vectors of the dimensionality of the state we estimate at node j. For example, with Figure 4b, the incoming messages would have dimension 16—one to modify the probability of each candidate high-resolution patch. Using $m_{ij}(x_j)$ to indicate the component of the vector $\mathbf{m}_{ij}$ corresponding to the patch candidate x_j, the rule[1,10] for updating the message from node i to node j is

$$m_{ij}\left(x_j\right) = \sum_{x_i} \phi_{ij}\left(x_i, x_j\right) \prod_{k \neq j} m_{ki}\left(x_i\right) \phi_i\left(x_i, y_i\right) \qquad (2)$$



7 Block diagram showing raster-order per-patch processing. At each step, we use local low- and high-frequency details (in green and red, respectively) to search the training set for a new high-frequency patch, which we add to the high-frequency image.

The sum is over all patch candidates x_i at node i, and the product is over all neighbors of the node i except for node j. Upon convergence, the belief-propagation estimate of the marginal probability $\mathbf{b}_i$ for each high-resolution patch x_i at each node i is

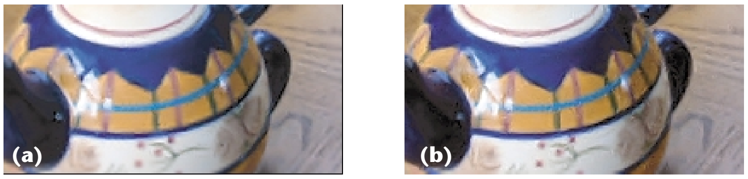
$$b_i\left(x_i\right) = \prod_k m_{ki}\left(x_i\right) \phi_i\left(x_i, y_i\right) \qquad (3)$$

(Yedidia, Freeman, and Weiss[11] show the connection between this estimate and an approximation used in physics due to Bethe. Freeman, Pasztor, and Carmichael[1] provide details of the belief-propagation implementation.)

One-pass algorithm

The fact that belief propagation converged to a solution of the Markov network so quickly led us to believe that simpler machinery might suffice. We found a one-pass algorithm that gives results that are nearly as good as the iterative solution to the Markov network.

In the one-pass algorithm, we only compute high-resolution patch compatibilities for neighboring high-resolution patches that are already selected, typically the patches above and to the left, in raster-scan order processing. If we prestructure the training data properly (see Figure 7), we can match the local low-resolution image data and select the compatible high-resolution patch candidate in a single operation—finding the nearest neighbor to a given input vector in the training set. The simplification avoids various steps in setting up and solving the Markov random field (MRF) of the previous section: finding the candidate set at each node, finding the compatibility matrices between all pairs of nodes, and using the iterative belief-propagation algorithm. Figure 8 shows a section of an image enlarged by both methods. We find that the one-pass algorithm is of approximately the same quality as the MRF-based algorithm for this problem.



8 **(a) Teapot image we enlarged by one octave using the belief-propagation method.[1] (b) Teapot image we enlarged by one octave using our one-pass method with the search method we describe in the "Search algorithm" section. The output of our simpler algorithm resembles that of the first.**

Algorithm details

In the simplest terms, one-pass super-resolution generates the missing high-frequency content of a zoomed image as a sequence of predictions from local image information. We subdivide the input image into low-frequency patches that are traversed in raster-scan order. At each step, a high-frequency patch is selected by a nearest neighbor search from the training set based on the local low-frequency details and adjacent, previously determined high-frequency patches.

As we already mentioned, it takes two steps to create an enlarged image with the desired number of pixels and corresponding additional image details. First, we double the number of pixels in the image, using a conventional image interpolation method such as cubic-spline or bilinear interpolation. Then, we predict missing image details in the interpolated image to create the super-resolution output.

In the algorithms we describe next, we perform the initial interpolation via cubic-spline interpolation. We scale an image down by convolving with a [0.25 0.5 0.25] blurring filter followed by subsampling on the even indices. (Freeman, Pasztor, and Carmichael[1] use linear interpolation for the upsampling, which puts slightly more interpolation burden on the rest of the algorithm.)

Prediction

Given the highest frequencies in an input image, the super-resolution algorithm predicts the next octave up—that is, the frequencies missing from an image zoomed with cubic interpolation. The algorithm's output is the sum of its input and the high-frequency predictions.

We predict the high frequencies for the $N \times N$ pixel patches in raster-scan order. Each prediction is based on two competing requirements. First, the high-frequency patch should come from a location in the training image that has a similar low-frequency appearance. Second, the high-frequency prediction should agree at the edges of the patch with the overlapping pixels of its neighbors to ensure that the high-frequency predictions are compatible with those of the neighboring patches.

We fulfill the first requirement by extracting a low-frequency patch—$M \times M$, not necessarily the same size as the high-frequency patch—from the image we're looking at to find a match in the training set, which is made up of pairs of low- and high-frequency patches. To meet the second requirement, we overlap predicted patches at their borders. When searching the training set, we also use the high-frequency data previously predicted to select the best pair. A user-controlled weighting factor α adjusts the relative importance of matching the low frequency patch versus matching the neighboring high-frequency patches in the overlap regions.

The super-resolution algorithm operates under the assumption that the predictive relationship between low- and high-resolution images is independent of local image contrast. Because of this, we normalize patch pairs by the average absolute value of the low-frequency patch across the color channels. (We add a small ε to avoid the denominator becoming zero at very low contrasts. ε effectively defines a floor of local image contrast below which we assume patch variability is due to noise.)

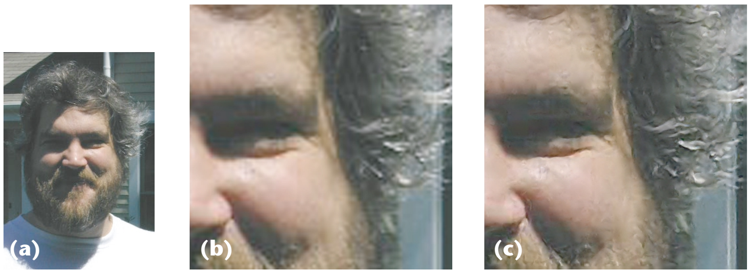
The pixels in the low-frequency patch and the high-frequency overlap are concatenated to form a search vector. The training set is also stored as a set of such vectors, so we search for a match by finding the nearest neighbor in the training set. When we find a match, we reverse the contrast normalization on the high-frequency patch and add it to the initial interpolation to obtain the output image (see Figure 7).

Search algorithm

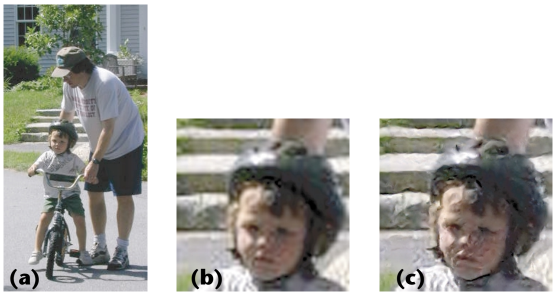
We search for matches using an L_2 norm. Due to the high dimension of the search space, finding the absolute best match would be computationally prohibitive. Instead, we use a tree-based, approximate nearest neighbor search. The tree is built by recursively splitting the training set in the direction of higher variation. At each step, we divide the set of tiles in half to maintain a balanced tree.

We use a best-first tree search to find a good match. This allows for a speed–quality trade-off: by searching more tree branches, we can find a better match. Because best-first search is unlikely to give the true best match without searching most or all of the tree, we improve the best-first match with a greedy downhill search in the graph connecting approximate nearest neighbors in the

9 Training images we used for the examples in this article (unless otherwise stated). We sampled patches at 1 pixel offsets over each of these images and over their synthetically generated low-resolution counterparts (after preprocessing steps). These six 200 × 200 images yielded a training set of slightly more than 200,000 high- and low-resolution image patch pairs.



10 (a) Original image. (b) Cubic-spline interpolation. (c) One-pass super-resolution interpolation.



11 Failure example. (a) Original image. (b) Cubic-spline interpolation by factor of 4 in each dimension. Note JPEG compression artifacts are visible. (c) One-pass super-resolution interpolation. Without high-level information, the algorithm treats the JPEG noise as a signal and amplifies it.

training set. This improves the match with negligible cost. In all one-pass algorithm examples in this article, we connect each patch pair to its 32 approximate nearest neighbors, which we compute with a method similar to Nene and Nayar.[12]

Training set and parameters

We build training sets for the super-resolution algorithm from band-pass and high-pass pairs taken from a set of training images. Spatially corresponding $M \times M$ low-frequency and $N \times N$ high-frequency patches are taken from image pairs.

Patch pairs are contrast normalized, as we described earlier. We create the search vector for a patch pair by concatenating the low-frequency patch and the region that will be overlapped in the high-frequency patch during the prediction phase, adjusted by the weighting factor α (see Figure 9).

We used the same set of training images for all the super-resolution examples in this article (see Figure 9). We took them with a Nikon Coolpix 950 digital camera at 640 × 480 resolution and used the highest quality compression settings.

Paying attention to parameter settings can improve image quality. For both levels of zooming, we used 5 × 5 pixel high-resolution patches (N = 5) with 7 × 7 pixel low-resolution patches (M = 7). The overlap between adjacent high-resolution patches was 1 pixel. These patch sizes capture small details well.

For a more conservative estimate of the higher resolution detail (not used here), we apply the algorithm four times at staggered offsets relative to the patch sampling grid. This gives four independent estimates of the high frequencies, which we can then average together, smoothing some image details but potentially reducing artifacts.

The parameter α controls the trade-off between matching the low-resolution patch data and finding a high-resolution patch that is compatible with its neighbors. The value

$$\alpha = 0.1 \frac{M^2}{2N-1}$$

gave good quality results in our experiments. The fraction compensates for the different relative areas of the low-frequency patches and overlapped high-frequency pixels as a function of M and N.

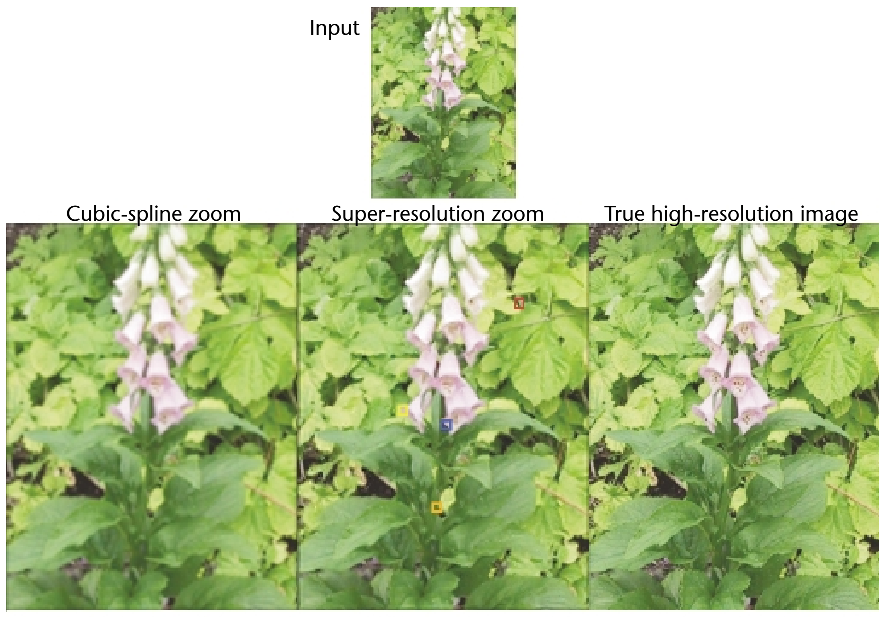
Results

Figure 10 shows our algorithm applied to a man's face. The training set is from the images in Figure 9. The resulting zooms are significantly sharper than those from cubic-spline interpolation, preserving sharp edges and image details.

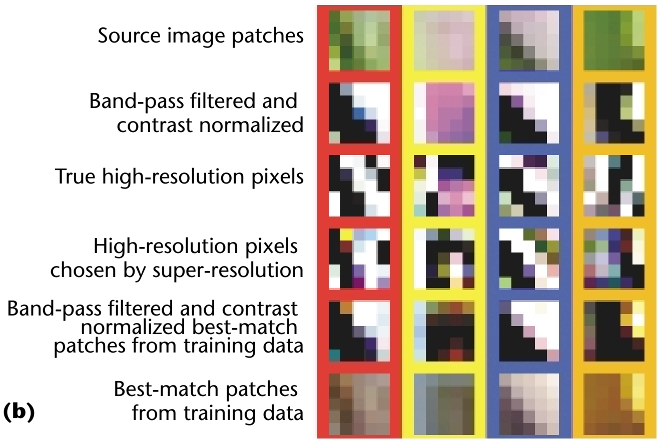
Figure 11 shows an example where our low-level training set alone isn't enough to distinguish JPEG compression noise from correct image data. The algorithm interprets the artifacts as image data and enhances them. Extensions of specialized high-level models[5] might properly handle images like this.

It might seem that to enlarge an image of one class—for example, a flower—we would need a training set that contained images of that same class—for example, other flowers. However, this isn't the case. Generic images can be a good training set for other generic images. Figure 12 shows an image (blurred and down-sampled from an original high-resolution image) zoomed with the one-pass super-resolution algorithm along with the same image zoomed with cubic spline and the original high-resolution image. Figure 12c shows the images we used from the training set in the super-resolution zoom. Figure 12b shows the details of a few patches in the zoomed image and their corresponding best matches in

(a)

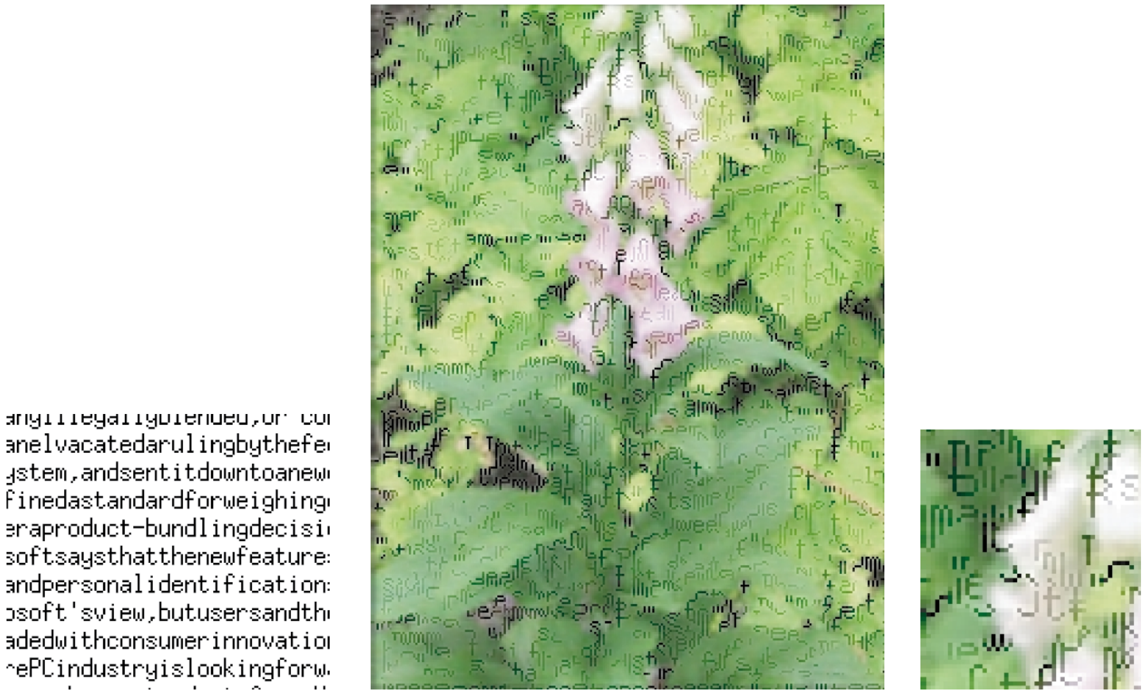


(b)


Source image patches
Band-pass filtered and contrast normalized
True high-resolution pixels
High-resolution pixels chosen by super-resolution
Band-pass filtered and contrast normalized best-match patches from training data
Best-match patches from training data

(c)

12 **Example showing how the (one-pass) algorithm uses patches in the training image to create detail in the test image. (a) Test image. (b) Patch matches. (c) Training images with location of best-match patches marked.**


anelvacatedarulingbythefe
ystem,andsentitdowntoanew
finedastandardforweighing
eraproduct-bundlingdecisi
softsaysthatthenewfeature
andpersonalidentification
osoft'sview,butusersandth
adedwithconsumerinnovatio
ePCindustryislookingforw

(a) (b) (c)

13 **Super-resolution example using a pathological training set composed entirely of text in one font. (a) An example image from the training set. (b) Zoomed image and (c) close-up. The algorithm does its best to invent plausible detail for this image, forming contours by concatenated letters.**

the training set. Figure 12b's top and bottom rows show the image content of the patches in the super-resolution image and the training set. The second and fifth rows in Figure 12b show the low-resolution, contrast normalized patches. Figure 12b's third row shows the high-resolution content of the original high-resolution image, and the fourth shows the high-resolution patch chosen by the super-resolution algorithm. Although not perfect, the matches between the original and estimated high-resolution patches are reasonably good. Note that the algorithm can use training patch examples from source image regions that look different than the regions where they are inserted into the zoomed image. For example, the orange bordered patch corresponds to a shadow boundary on wood in the training image (of three girls), but the algorithm applies it to zoom up a green plant occlusion boundary. The band-pass filtering and contrast normalization allows for this reuse, which makes the training set more powerful.

Although the training set doesn't have to be very similar to the image to be enlarged, it should be in the same image class—such as text or color image. In Figure 13, we enlarged the image in Figure 12a using a pathological training set of images of text. (Freeman, Pasztor, and Carmichael[1] give related experiments.) The algorithm does its best to explain the observed low-resolution image in its vocabulary of text examples, resulting in a zoom with high-resolution detail formed out of concatenated characters.

Discussion and conclusions

Recent patch-based texture synthesis models[13,14] also use spatial consistency constraints similar to those we applied here. Our method differs from that of Hertzmann et. al[4] because it operates on tiles rather than per-pixel, providing a performance benefit. It also normalizes the training set according to contrast and assumes that the highest frequency details in an image can be predicted using only the next lower octave. These two generalizations let us enlarge a wider class of images using a single, generic training set, rather than restricting us to operating on images that are very similar to the training image. Pentland and Horowitz[15] used a training-based approach to super-resolution, but they didn't use the spatial consistency constraints that we believe are necessary for good image quality.

If well-known objects are sparsely sampled in the image, an image extrapolation based on local image evidence alone won't produce the new details that the viewer expects. Very small face images are susceptible to this problem. To address these properly, we would have to add higher level reasoning to the algorithm. (See Baker and Kanade[5] or Liu, Shum, and Zhang[6] for super-resolution algorithms tuned to a particular class of images, such as faces or text.)

In the zoomed-up images, low-contrast details next to high-contrast edges can be lost because of the contrast normalization fixing on the level of the high-contrast edge. Independent contrast normalization for different image orientations, each zoomed separately, might address this problem. However, it isn't clear that a one-pass implementation would suffice for that modification.

Finally, our algorithm works best when the data's resolution or noise degradations match those of the images to which it's applied. Numerically, the root-mean-squared error from the true high frequencies tend to be approximately the same as for the original cubic-spline interpolation. Unfortunately, this metric has only a loose correlation with perceived image quality.[16] Typical processing time for the single-pass algorithm is 2 seconds to enlarge a 100×100 image up to 200×200 pixels.

We've focused on enlarging single images. Enlarging moving images is different in two respects. More input data exists, so multiple observations of the same pixel could be used for super-resolution. Also, we must take care to ensure coherence across subsequent frames so that the made-up image details don't scintillate in the moving image.

Our algorithms are an instance of a general training-based approach that can be useful for image-processing or graphics applications. Training sets can help enlarge images, remove noise, estimate 3D surface shapes, and attack other imaging applications. ■

Acknowledgments

We thank Ted Adelson, Owen Carmichael, and John Haddon for helpful discussions.

References

1. W.T. Freeman, E.C. Pasztor, and O.T. Carmichael, "Learning Low-Level Vision," *Int'l J. Computer Vision*, vol. 40, no. 1, Oct. 2000, pp. 25-47.
2. W.T. Freeman and E.C. Pasztor, "Learning to Estimate Scenes from Images," *Adv. Neural Information Processing Systems*, M.S. Kearns, S.A. Solla, and D.A. Cohn, eds., vol. 11, MIT Press, Cambridge, Mass., 1999, pp. 775-781.
3. W.T. Freeman and E.C. Pasztor, "Markov Networks for Superresolution," *Proc. 34th Ann. Conf. Information Sciences and Systems* (CISS 2000), Dept. Electrical Eng., Princeton Univ., 2000.
4. A. Hertzmann et al., "Image Analogies," *Computer Graphics* (Proc. Siggraph 2001), ACM Press, New York, 2001, pp. 327-340.
5. S. Baker and T. Kanade, "Limits on Super-Resolution and How to Break Them," *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (CVPR), vol. II, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 372-379.
6. C. Liu, H. Shum, and C. Zhang, "A Two-Step Approach to Hallucinating Faces: Global Parametric Model and Local Non-Parametric Model," *Proc. Int'l Conf. Computer Vision* (ICCV), vol. I, IEEE CS Press, Los Alamitos, Calif., 2001, pp. 192-198.
7. D.J. Field, "What Is the Goal of Sensory Coding," *Neural Computation*, vol. 6, no. 4, July 1994, pp. 559-601.
8. O. Schwartz and E.P. Simoncelli, "Natural Signal Statistics and Sensory Gain Control," *Nature Neuroscience*, vol. 4, no. 8, Aug. 2001, pp. 819-825.
9. S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no. 4, Nov. 1984, pp. 721-741.
10. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Francisco, 1988.
11. J.S. Yedidia, W.T. Freeman, and Y. Weiss, "Generalized Belief Propagation," *Advances in Neural Information Processing Systems*, T.K. Leen, T.G. Dietterich, and V. Tresp, eds., vol. 13, MIT Press, Cambridge, Mass., 2001, pp. 689-695.
12. S.A. Nene and S.K. Nayar, "A Simple Algorithm for Nearest Neighbor Search in High Dimensions," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 9, Sept. 1997, pp. 989-1003.
13. A.A. Efros and W.T. Freeman, "Image Quilting for Texture Synthesis and Transfer," *Computer Graphics* (Proc. Siggraph 2001), ACM Press, New York, 2001, pp. 341-346.
14. L. Liang et al. "Real-Time Texture Synthesis by Patch-Based Sampling," *ACM Trans. Graphics*, vol. 20, no. 3, July 2001, pp. 127-150.
15. A. Pentland and B. Horowitz, "A Practical Approach to Fractal-Based Image Compression," *Digital Images and Human Vision*. A.B. Watson, ed., MIT Press, Cambridge, Mass., 1993.
16. W.F. Schreiber, *Fundamentals of Electronic Imaging Systems*, Springer-Verlag, New York, 1986.

William T. Freeman *is an associate professor of computer science at the Massachusetts Institute of Technology in the Artificial Intelligence Lab. His research interests include machine learning applied to computer vision and computer graphics, Bayesian models of visual perception, and interactive applications of computer vision. He has a PhD in computer vision from MIT and worked at Mitsubishi Electric Research Labs for nine years.*

Thouis R. Jones *is a graduate student in the Computer Graphics Group at the MIT Laboratory for Computer Science. His research interests include shape representation for computer graphics, antialiasing, and super-resolution. He has a BS in computer science from the University of Utah.*

Egon C. Pasztor *is a graduate student at the MIT Media Lab, where he has worked on computer vision and is currently working on interfaces for computer-assisted musical composition. His research interests include human–computer interfaces and interaction technologies that make machine interaction a more natural and productive experience. He has a BS in computer science from California Institute of Technology.*

Readers may contact William Freeman at the MIT Artificial Intelligence Lab, 200 Technology Square, Cambridge, MA 02139, email wtf@ai.mit.edu.