Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Image and Vision Computing 24 (2006) 1039–1052

image AND vision COMPUTING

www.elsevier.com/locate/imavis

Image super-resolution survey

J.D. van Ouwerkerk *

Department of Media and Knowledge Engineering, Delft University of Technology, Mekelweg 4, 2628 CL Delft, The Netherlands

Received 23 May 2005; received in revised form 8 December 2005; accepted 21 February 2006

Abstract

The shortcomings in commonly used kernel-based super-resolution drive the study of improved super-resolution algorithms of higher quality. In the past years a wide range of very different approaches has been taken to improve super-resolution.

This paper compares approaches to high quality super-resolution by looking at theoretical backgrounds and practical results. Strengths and weaknesses are listed with the intent to spot chances for combination or improvement of techniques, thereby forming a base for future improved super-resolution algorithms.

Keywords: Super-resolution; Single frame

1. Introduction

Super-resolution is the process of generating a raster image with a higher resolution than its source. The source can consist of one or more images or frames. This paper focuses on single-frame super-resolution, meaning that the source is a single raster image. Single-frame super-resolution is also known as image scaling, interpolation, zooming and enlargement.

Resampling of images to change size, resolution or orientation is common in all sorts of devices, like computers, television sets, mobile phones and digital camera's. Performing this resampling by using kernels is easy and effective, but not always optimal in terms of quality. Specific algorithms have been designed for scaling up, scaling down and rotating images that deliver higher quality results. This paper focuses on the process of scaling up or super-resolution, which in contrast to scaling down or rotating deals with the problem of increasing the amount of pixels or, more general, data.

Super-resolution is among others applied to enlarge digital (consumer) photographs, increase printer resolution and convert PAL or NTSC resolution video to HDTV resolution video. Like resampling in general, super-resolution is a common process that is performed frequently by devices like digital camera's and computers.

The classical way of obtaining super-resolved images is by using kernels as described in Section 2.1. The implementation of this approach is easy, but the results are not always as good as one envisions. Some shortcomings of the kernel-based approach are apparent and can be attributed to the underlying model of the interpolated signal. These three shortcomings are identified next and will have a central role in the subjective evaluation.

A common problem with kernel super-resolution is the blurring of sharp edges. Kernel filters typically perform very well in smooth areas, but not in edge areas. Fig. 1 shows a graphical image at the top left and at the bottom left the same image after decimation and super-resolution by one octave (a factor of two) using a linear kernel. The intensities of a 10 pixel part are displayed in the graph on the right of the image. The darker line represents the original image, while the lighter line represents the super-resolved version. It is clear that the originally steep edge has become less steep in the super-resolved image, which is visible as edge blurring.

The second problem is the introduction of blocking artifacts in diagonal edges or lines as shown in Fig. 2.

The diagonal line in this image not only looks blurry, but a staircase pattern is also emerging in the bilinearly super-resolved image. This staircase pattern or blocking artifacts is caused by the horizontal and vertical orientation of the resampling kernels. Kernel super-resolution is unable to recognize or follow diagonal lines, which causes blocking.

The third problem is the inability to generate high frequency components or fine detail, as shown in Fig. 3. This is needed to make the super-resolved image look more plausible.

* Tel.: +31 6 1661 2001.
E-mail address: jos@jvojava.com

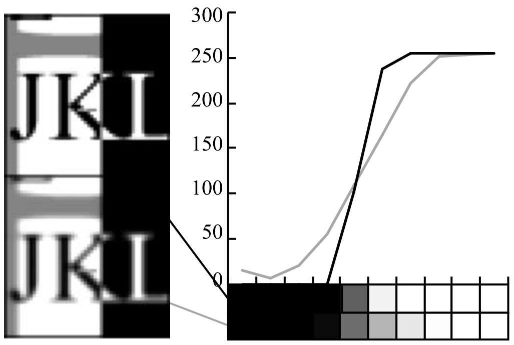

Fig. 1. Original and result after decimation and bilinear super-resolution by one octave (factor of two).

The original and super-resolved image are taken through low pass and high pass filters to show the higher deterioration in the high pass signal compared to the low pass signal.

These shortcomings have driven the study of improved super-resolution algorithms that address some or all of these problems trying to increase the quality of the super-resolved images. The approaches taken in the past few years sometimes differ from each other greatly. Some use explicit models to represent natural images, while others use machine learning to obtain the relation between a source image and its super-resolved counterpart. Some use the concept of pixel classes to perform class specific super-resolution for each pixel, while others use a single algorithm for every pixel in the image. Some use iterative belief propagation to gradually improve the super-resolved image, while others perform super-resolution in a single pass.

What they do have in common is that they take advantage of the strong relation between neighbouring pixels to estimate values of missing pixels. They also have in common that they treat the image as a two-dimensional signal instead of separating the image into a range of one-dimensional signals

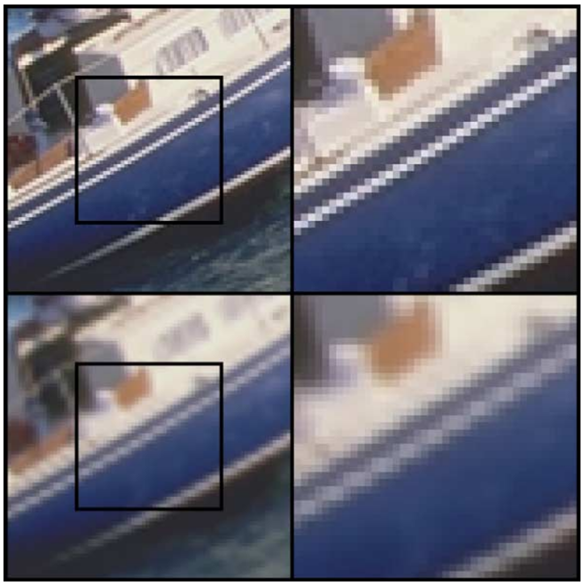

Fig. 2. Original and result after decimation and bilinear super-resolution by one octave.

to take advantage of the two-dimensional patterns occurring in natural images.

In the past few years, a whole range of super-resolution techniques and algorithms were introduced. This paper aims to shine some light on these techniques and algorithms by describing them and comparing them by looking at theoretical backgrounds and practical results. The theoretical view aims at determining how each algorithm reduces blurring and blocking and increases detail. The test results are used to determine how effective blurring and blocking is reduced and detail is increased. Strengths and weaknesses of each algorithm are listed with intend to form a base for future improved super-resolution algorithms.

The remainder of this paper is organized as follows. In Section 2 a survey of super-resolution algorithms is presented. Section 3 describes the test setup. In Section 4 test results are listed and Section 5 concludes the paper.

2. Algorithms

Freeman et al. approached super-resolution from a low level vision learning perspective. An approach to low level vision tasks using belief propagation is presented in [12]. The scene underlying the supplied image data is estimated using a Markov network. To make the estimation feasible, both the image data and the scene are separated into patches. This approach to low level vision is specifically applied to super-resolution in [13]. It uses the high frequency part of the low resolution image as ground truth and the high frequency part of the high resolution image as scene to be estimated. A variant of this algorithm incorporating the distribution of pixel intensity derivatives is presented in [14]. A faster version of the algorithm that only uses a single pass is introduced in [8]. In this paper, the belief propagation technique is described in Section 2.6.

Atkins et al. approached super-resolution using pixel classification. Pixel classification aims to sort pixels into classes like horizontal edges and smooth areas. A tree-based classification approach to super-resolution is introduced in [15]. This algorithm builds a decision tree with a linear interpolator at each leaf of the tree. Each non-leaf node in the tree represents a binary choice. In [1] a similar algorithm is introduced, which assigns a pixel to one or more classes in a single step instead of using a set of binary choices. The algorithm introduced in [1] is described in Section 2.2 of this paper and included in the test results.

Battiato et al. have published papers on several super-resolution algorithms. A rule based super-resolution approach called LAZA is described in [5]. In LAZA, the authors use simple rules and configurable thresholds to detect edges and update the interpolation process accordingly. In [9] the same authors introduce an algorithm that incorporates anisotropic diffusion (SIAD) to sharpen edges. The SIAD algorithm is described further in Section 2.5, while the LAZA algorithm is described in Section 2.8. Both LAZA and SIAD are included in the test results. In [16] Battiato et al. compare several super-

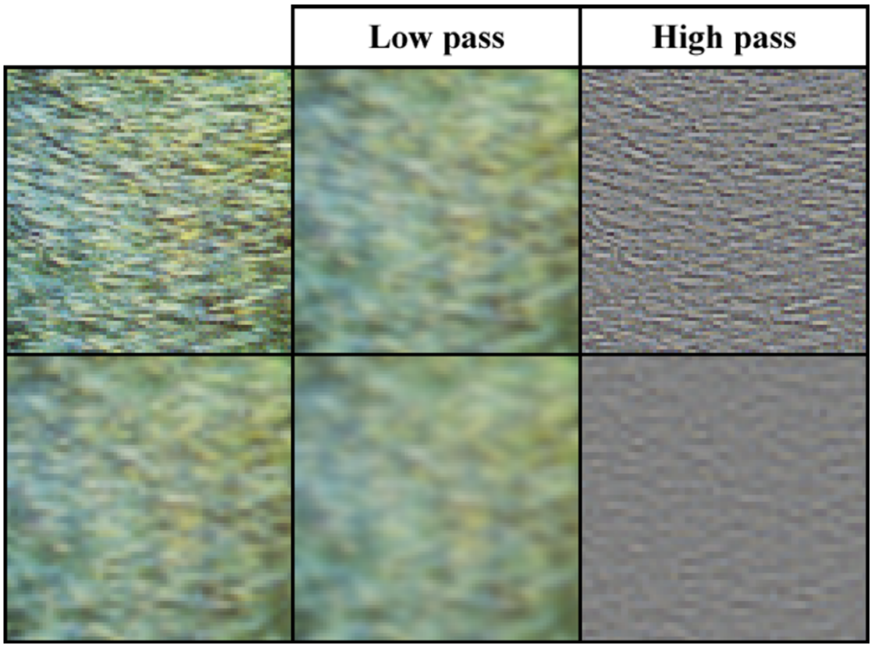

Fig. 3. Original and result after decimation and bilinear super-resolution by one octave.

resolution algorithms including LAZA and SIAD using the PSNR measure described in Section 3.2.1.

Muresan and Parks published several papers [3,10,17–19] on super-resolution based on the optimal recovery principle. The authors model the image as belonging to a certain ellipsoidal signal class. Together with Kinebuchi a wavelet-based algorithm using hidden Markov trees was introduced in [20]. It uses lower frequency wavelet coefficients to predict the highest frequency coefficients. By applying an inverse wavelet transform after prediction, an one octave super-resolved image results.

Su and Willis present super-resolution by triangulation on pixel level in [11]. Yu et al. present super-resolution by data-dependant triangulation (DDT) in [26]. These methods use linear interpolation that is not generally aligned with the coordinate axes to reduce visible artifacts caused by this alignment. It is shown by the last authors that results can be further improved by improving the algorithm that searches for the optimal triangulation of the source image.

Another approach that is explicitly directed at maintaining sharp edges is the use of Subpixel edge localization by Jensen and Anastassiou [27]. This approach detects the most prominent edge in the local window with subpixel precision and uses the resulting edge template to obtain sharper edges in super-resolved images.

Other techniques include training a neural network for interpolation using a spatial error measure as proposed by Staelin et al. [2] and described in Section 2.3. Li and Orchard propose new edge-directed interpolation (NEDI) in [6], which makes use of the geometric duality between the covariance in the low and high resolution images. This algorithm has been extended by Muresan and Parks as noted in [3] and is described in Section 2.7. Xu et al. propose super-resolution using a combination of wavelet and fractal image models in [7].

There are also a range of commercial products available that rely on an algorithm more advanced than kernel-based resampling. I would like to mention PhotoZoom Professional by BenVista [28], Imagener by Kneson Software [29], Qimage by Digital Domain [30], Pictura by Digital Multi-Media Design [31] and SmartScale by Extensis [32]. The Pictura software makes use of a modified version of the algorithm presented in [3] by Muresan and Parks. The PhotoZoom Professional software was previously known as S-Spline by Shortcut, but has gone through some changes in name and company.

2.1. Kernel resize

The most common way of achieving super-resolution is using a base function or interpolation kernel (KERN). This algorithm uses the base function to approximate the continuous function underlying the discrete samples that make up the image. The continuous function is approximated by combining instances of the base or kernel function φ multiplied by the known discrete samples f_i for all pixel locations i in Z and is a linear operation [21] as shown in Eq. (1). Note that the equation in (1) is for the one-dimensional case:

$$f(x) = \sum_{i \in Z} f_i \cdot \varphi(x - i) \tag{1}$$

An example approximation using a Lanczos kernel with radius 3 is shown in Fig. 4. The thin vertical lines denote the known discrete samples and the thick line denotes the continuous approximation.

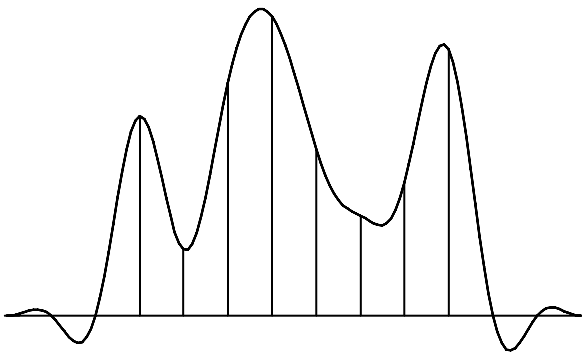
Fig. 4. Continuous approximation using a Lanczos kernel.

Typical choices of base functions include linear, cubic spline and Lanczos, which are shown in Fig. 5.

Since the used base functions are continuously valued functions, there is no limit on the scaling factor. Kernels used for super-resolution are typically linearly separable so the process can be split in two consecutive steps, horizontal and vertical, to reduce computation. This reduces the process to a single dimension instead of treating the data two-dimensional.

Interpolation using a linear kernel is also known as bilinear interpolation and interpolation using a cubic kernel is also known as bicubic interpolation. Eq. (2) denotes the one-dimensional Lanczos base function with radius r:

$$\varphi(x) = \begin{cases} \dfrac{r \sin(x\pi) \cdot \sin(x\pi/r)}{x^2\pi^2}, & |x| < r \\ 0, & |x| \geq r \end{cases} \tag{2}$$

2.2. Resolution synthesis

Resolution synthesis (RS) super-resolution as described in [1] is based on the assumption that there are different classes of pixels, like classes of pixels on object edges with different orientations and the class of pixels in flat areas, that each require specific treatment when enlarged. Each of these classes can benefit from a dedicated super-resolution scheme to better preserve edges and generate details. The class of pixels on horizontal edges for example need to be treated in a way that doesn't blur the present horizontal edge. Clustering, also known as unsupervised learning, groups similar inputs into a number of classes or clusters.

In RS each pixel can be assigned to multiple classes with different degrees. A pixel can for example be classified to be 60% horizontal edge and 40% smooth. For each pixel and class the chance of membership is estimated using a Gaussian distribution. The centre of the Gaussian distribution is positioned at the centre of the pixel class. The chances are later normalised for each pixel to sum up to one.

The algorithm uses a specific linear filter for each class. A filter window containing the local neighbourhood of a low resolution pixel is constructed. This filter window is used as input for the linear filter to form the high resolution pixels.

A projection operator constructs a classification window from the filter window. This classification window is typically smaller and has the centre pixel subtracted. It is then scaled to accentuate edges. The classification window is used to determinate the degree of membership to each pixel class. The degree of membership to a class equals the chance of the low resolution pixel belonging to this class based on Gaussian distributions. These chances act as weights for the results of the linear filters for each class. The weighted average of the filter results is the final result.

Let n be the number of inputs into the linear filter and c the number of classes. Let I be the vector of size n containing the input values from the filter window, C the matrix of size $n \times c$ containing the filter coefficients for all classes and M the vector of size c containing the memberships to each class. The output value o is calculated using Eq. (3):

$$o = I^{\mathrm{T}} \cdot (C \cdot M) \tag{3}$$

The RS algorithm is limited to an integer scaling factor. It attempts to use classification into edge and non-edge pixels with different orientations to limit edge blurring and blocking effects.

The implementation of RS in [1] uses eight difference values between the centre pixel and its direct neighbours for classification. The input of the linear filter is a 5×5 pixel window around the low resolution pixel. The number of classes used is 100. It is advised to use at least 100,000 training examples.

2.3. Spatial neural network

Neural network image scaling using spatial errors (NNSE) as described in [2] uses a single generalized feed forward neural network as interpolator.

The input of the neural network consists of the pixels in the local window around the source pixel and the output consists of the pixels needed for the super-resolved image. The neural network can be trained by using a decimated image as input and the corresponding original image as target.

The value of the centre pixel in the input window is subtracted from the inputs and is later added to the outputs. This input normalisation reduces the difference between cases

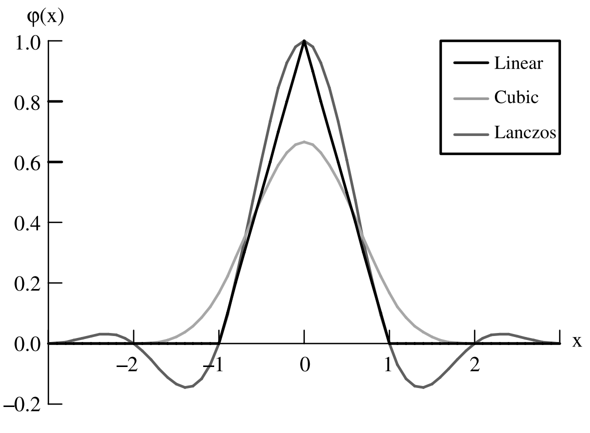

Fig. 5. Typical super-resolution kernel functions.

to improve the training process, while staying general. The contrast in the input window is also normalised to further improve training.

While kernel resize is always linear, a multilayer neural network can be trained to recognize non-linear relations between input and output. It is, therefore, able to better preserve edges and enhance detail than linear interpolators.

In the generalized feedforward network used, the input of each hidden node consists of the input nodes and all previous hidden nodes. The input for the output nodes consists of all hidden. The first input node is a bias node with a constant value of one.

The training error introduced into the neural network is a spatial error measure instead of the basic squared error. The spatial error measure combines the squared errors in a 3×3 local window. Let the errors in the 3×3 window be stacked column by column in vector V of size 9. A 9×9 matrix A is designed to accentuate errors on edges and lines. The spatial error e is calculated using Eq. (4).

$$e = V^{\mathrm{T}} \bullet A \bullet V \tag{4}$$

Matrix A is composed of the nine 3×3 kernels shown in Fig. 6. The first kernel is an average kernel, the next four are edge detection kernels and the last four are line detection kernels. Each kernel in Fig. 6 is normalised and weighted before forming a row in matrix A.

The implementation of NNSE in [2] uses a feed forward network with 30 hidden nodes, a tanh activation function and a 5×5 pixel input window. The centre pixel is subtracted from the window and the contrast in the window is scaled to accentuate edges.

2.4. Local correlation

Local correlation super-resolution (LCSR) as described in [4] is a two step algorithm. The first step is a classification step and the second step consists of assigning a local associative memory (LAM) to each class of pixels. Each LAM is trained using a single step or gradient descent depending on the number of layers in the LAM.

LCSR is very similar to RS as described in Section 2.2. The main difference is that in LCSR each pixel belongs to a single class during super-resolution, while in RS each pixel can belong to multiple classes. LCSR uses a local input window like RS, which is used as input to the LAM and the classification.

$$\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\begin{bmatrix}-1&-1&-1\\0&0&0\\1&1&1\end{bmatrix}\begin{bmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{bmatrix}$$
$$\begin{bmatrix}0&1&1\\-1&0&1\\-1&-1&0\end{bmatrix}\begin{bmatrix}-1&-1&0\\-1&0&1\\0&1&1\end{bmatrix}\begin{bmatrix}-1&-1&-1\\2&2&2\\-1&-1&-1\end{bmatrix}$$
$$\begin{bmatrix}-1&2&-1\\-1&2&-1\\-1&2&-1\end{bmatrix}\begin{bmatrix}2&-1&-1\\-1&2&-1\\-1&-1&2\end{bmatrix}\begin{bmatrix}-1&-1&2\\-1&2&-1\\2&-1&-1\end{bmatrix}$$

Fig. 6. Nine 3×3 spatial error kernels.

The training exists of two steps. In the first step pixel, classification is performed using a Kohonen self-organising map (SOM) using the local window as input. The second step consists of training each LAM for the pixels that fall into the corresponding class.

The implementation of LCSR in [4] uses a 3×3 or 5×5 input window and up to 30 classes. The mean of the input window is normalised to zero.

2.5. Anisotropic diffusion

Smart interpolation by anisotropic diffusion (SIAD) as described in [9] uses anisotropic diffusion to sharpen edges and generate plausible detail. Anisotropic diffusion lets pixel intensity values diffuse over neighbours. The diffusion at a point is inversely proportional to the local contrast to enhance edges. Anisotropic diffusion is able to estimate a piecewise smooth image from a noisy source as noted in [22].

To reduce artefacts and aliasing caused by the anisotropic diffusion, the image is reduced in size afterwards. This requires the image to be enlarged beyond the needed size before the anisotropic diffusion step.

Super-resolution using SIAD is a three step process. The first step consists of enlarging the image beyond the required resolution using kernel resize. The enlargement factor used in [9] is eight. This will create a large, but blurry image. The second step consists of performing the anisotropic diffusion. This will sharpen the blurry edges. The third and final step consists of reducing the image by averaging pixels. This is done with a factor four to obtain an image super-resolved by one octave.

2.6. Sparse derivative prior

Super-resolution exploiting the sparse derivative prior (SDPSR) as described in [14] uses the distribution of the derivative of natural images as model. This distribution of this derivative or differences between neighbouring pixels is sharply peaked at zero for natural images and is modelled with a generalized Laplace distribution shown in Eq. (5). The graph of this function for the range -100 to 100 and $s=20$ for different values of α is shown in Fig. 7:

$$p(x) = \exp(-|x|^{\alpha}/s) \tag{5}$$

Every low resolution neighbourhood is interpolated by the same set of interpolators to form a set of high resolution patches for each location. The selection of patches is updated iteratively maximizing the chance of the selection using belief propagation (BP). The choice of a patch is influenced by the low resolution image and the choice of neighbouring patches, creating a wider spatial dependency of high resolution pixels with each iteration. The BP in SDPSR uses two types of constraints, a generalized Laplace distribution for differences between neighbouring pixels and a Gaussian distribution for the differences between the low resolution pixel and the corresponding high resolution patch. While it is not guaranteed

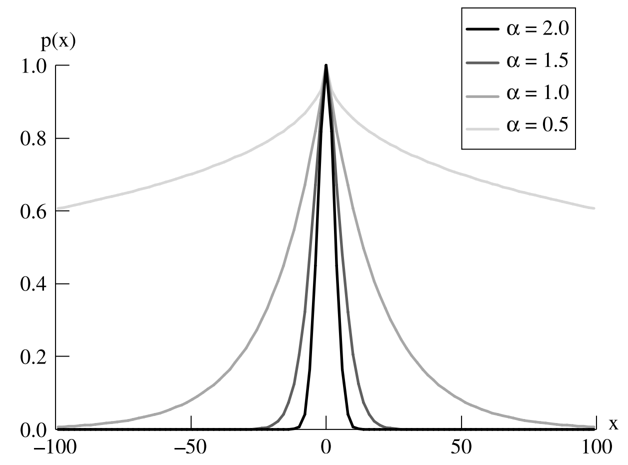

Fig. 7. Generalized Laplace distribution for different values of α.

that belief propagation containing loops will converge, it normally does when applied to natural images. Satisfactory results have been obtained in any case.

The implementation of SDPSR in [14] trains a set of interpolators by clustering low resolution pixel neighbourhoods into 16 classes and calculating the RMS-optimal linear interpolator for each class. During super-resolution each of these interpolators is applied to each low resolution pixel and the optimal one is selected by BP afterwards. About 500,000 training vector pairs were used for training. The BP needed five iterations in the experiments done in [14].

2.7. Edge-directed

New edge-directed interpolation (NEDI) as described in [6] uses the duality between the low resolution and high resolution covariance for super-resolution. The covariance between neighbouring pixels in a local window around the low resolution source is used to estimate the covariance between neighbouring pixels in the high resolution target. An example covariance problem is represented in Fig. 8.

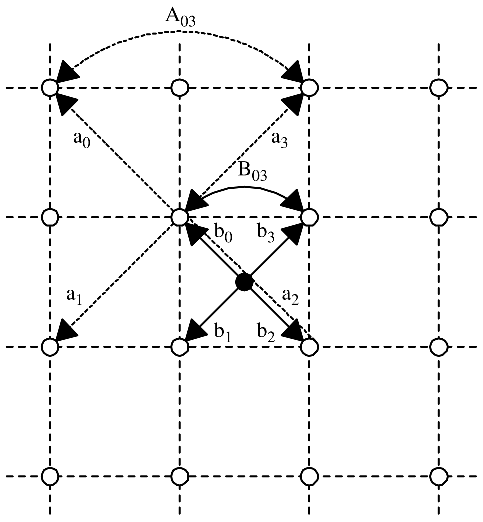

Fig. 8. Local covariance.

The covariance of the b_0 relation in Fig. 8 is estimated by the covariance of neighbouring a_0 relations in the local window. The open circles in the figure represent the low resolution pixels and the closed circle represents a high resolution pixel to be estimated.

The covariance used is that between pixels and their four diagonal neighbours. The covariance between low resolution pixels and their four diagonals in a $m \times m$ local window is calculated. This covariance determines the optimal way of blending the four diagonals into the centre pixel. This optimal value in the low resolution window is used to blend a new pixel in the super-resolution image. By using the local covariance, the interpolation can adhere to arbitrarily oriented edges to reduce edge blurring and blocking.

Let I be a vector of size four containing the diagonal neighbours of the target pixel o, X a vector of size m^2 containing the pixels in the $m \times m$ window and C a $4 \times m^2$ matrix containing the diagonal neighbours of the pixels in X. The equation for enlargement using this method is shown in (6):

$$o = (C^{\mathrm{T}} \cdot C)^{-1}(C^{\mathrm{T}} \cdot X) \cdot I^{\mathrm{T}} \tag{6}$$

The NEDI algorithm uses two passes to determine all high resolution pixels. The first pass uses the diagonal neighbours to interpolate the high resolution pixels with both coordinates odd and the second pass uses the horizontal and vertical neighbours to interpolate the rest of the high resolution pixels as shown in Fig. 9.

2.8. Locally-adaptive

The locally-adaptive zooming algorithm (LAZA) introduced in [5] uses a set of simple rules to take information about discontinuities or sharp luminance variations into account while doubling the horizontal and vertical resolution of the input image. The base scaling factor of the LAZA algorithm is thereby 2×2, but by repeating the process, any power of two scaling factor can be obtained.

The algorithm is performed in four steps. All steps except the last one can leave pixel values undefined. In the case that they are left undefined, they will be set in a later step. The first step simply spreads out the pixel values to locations with two odd coordinates as is shown in the two leftmost steps of Fig. 10(b). The second and third steps set the values of undefined pixels that are detected to have a 'simple' spatial

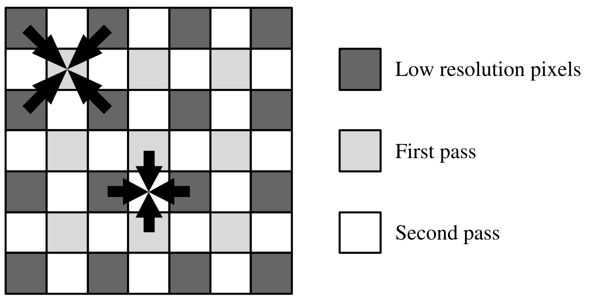

Fig. 9. Two steps in NEDI.

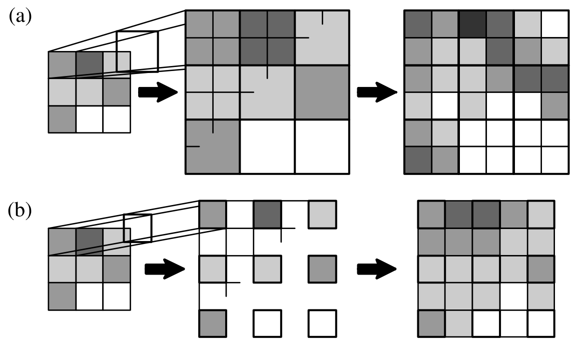

Fig. 10. (a) Even parity; (b) odd parity.

dependence with the neighbouring pixels. The membership to a spatial dependency is determined by a simple rule that uses surrounding pixel values and two threshold values. The second step recognizes five spatial dependencies, while the third step recognizes four spatial dependencies. After the first three steps, some pixels might still be undefined. All undefined pixels will, therefore, be set in the fourth and final step. In this final step, four surrounding pixels of the undefined pixel are combined to form the value of this pixel. To preserve more detail, the four pixel values are not simply averaged to form the new value. The value spectrum is instead divided into a number of bins. The four values of the surrounding pixels are placed into these bins and the values of the bins with at least one pixel value in them are averaged. The median of the values in the bin is usually taken as the value of the bin.

2.9. Overview

There are some things that all or most algorithms have in common. One important one is that they always use a local low resolution window as input. This local window often has a size of 3×3 or 5×5. The centre pixel is often subtracted from all values in the window (offset) and in some cases the contrast in the window is scaled to accentuate edges (scale). The used normalizations are listed in Table 1.

Most algorithms are limited to an integer scaling factor; unlike kernel resize, which has no limit on the scaling factor. NEDI and LAZA are limited to a scaling factor of two. A limit on the scaling factor does not mean that other scaling factors cannot be obtained. An algorithm can be repeated until the resulting image is at least as large as the required size. The resulting image can then be scaled down to the required size using kernel resize for example. Limits on scaling factors are listed in Table 1.

Table 1
Algorithm overview

Paragraph	Code	Normalisation	Scale factor	Parity	Trained
2.1	KERN	None	Any	Any	No
2.2	RS	Offset	Integer	Even	Yes
2.3	NNSE	Offset+scale	Integer	Even	Yes
2.4	LCSR	Offset	Integer	Even	Yes
2.5	SIAD	None	Integer	Any	No
2.6	SDPSR	None	Integer	Odd	Yes
2.7	NEDI	None	2	Odd	No
2.8	LAZA	None	2	Odd	No

Each algorithm either divides a low resolution pixel into several high resolution pixels or generates extra pixels between low resolution pixels. The difference in these two approaches is shown in Fig. 10 for super-resolution by one octave.

Fig. 10(a) demonstrates super-resolution with even parity enlarging a $n \times m$ image to a $2n \times 2m$ image. Every low resolution pixel is divided into four high resolution pixels. Fig. 10(b) demonstrates super-resolution with odd parity enlarging a $n \times m$ image to a $(2n-1) \times (2m-1)$ image. The low resolution pixels are first moved apart and pixels are added in between them to form the high resolution image. The term super-resolution parity is chosen, because it coincides with the parity of the number of high resolution pixels after super-resolution by one octave.

Four algorithms have to be trained by 'showing' example images. These four are RS, NNSE, LCSR and SDPSR. Training is used both supervised to train interpolators and unsupervised to perform clustering.

3. Test setup

Both objective and subjective tests are performed. Both tests make use of the same set of test images. The objective test will make use of several objective measures, while the subjective test is aimed at performance in edge blurring, edge blocking and generation of detail.

Error measures are used to objectively compare a super-resolution image to the original one as shown in Fig. 11. The original image is first decimated by factor f and then super-resolved with factor f. The original and super-resolved images are compared using an error measure.

The decimation used in this paper consists of applying a block or zero order filter followed by a down sampling step. For even decimation by one octave, this comes down to taking the average of four high resolution pixels for each low resolution pixel.

3.1. Test images

The test set consists of the seven images with an original resolution of 512×512 shown in Fig. 12. The images have been selected to test on edge blurring, edge blocking and generation of detail.

The names of the images in Fig. 12 from left to right and top to bottom are graphic, lena, mandrill, monarch, peppers, sail and tulips. The graphic image has been made for this test. The other images are royalty free and have been used in other computer graphics performance tests. Some, like lena and peppers, are widely used for performance tests in computer graphics.

The graphic and monarch images exhibit strong and sharp edges in varying directions, making them prime images to test edge blurring and edge blocking effects. The mandrill image contains a lot of detail in the hairs, making it a prime image to test detail generation. A sharp diagonal white line is visible on

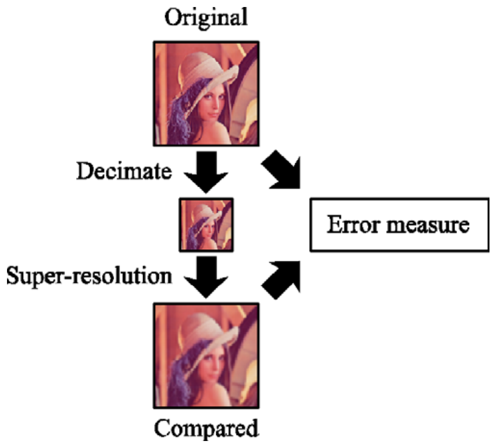

Fig. 11. Error measure layout.

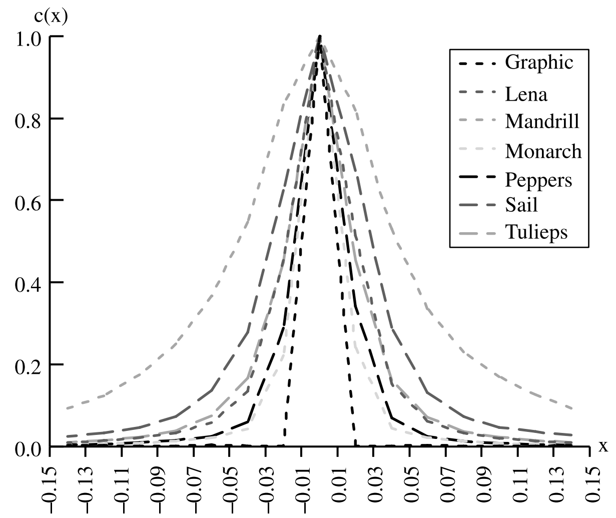

Fig. 13. Horizontal first derivative histogram.

the side of the boat in the right of the sail image. This makes the super-resolved sail image prone to display edge blocking.

The first derivative of an image gives some insight in the amount of detail present in the image. Part of the histograms of the horizontal first derivative of the seven test images are shown in Fig. 13. The images are first converted to grayscale with Eq. (7). The grayscale intensity value y is assumed to be a linear combination of the red, green and blue intensity values r, g and b:

$$y = 0.299r + 0.587g + 0.114b \tag{7}$$

Then each difference d in grayscale intensities y of two horizontally neighbouring pixels is calculated. These differences are grouped into bins that are 0.02 wide. The counts of elements in each bin are shown in Fig. 13 after normalisation for each image.

The derivative histograms are sparse as noted and used explicitly for super-resolution in [14]. It can be seen that the mandrill image has the least sparse derivative histogram, since it contains the most detailed texture. The graphic image has the sparsest derivative histogram, since it does not contain any texturing.

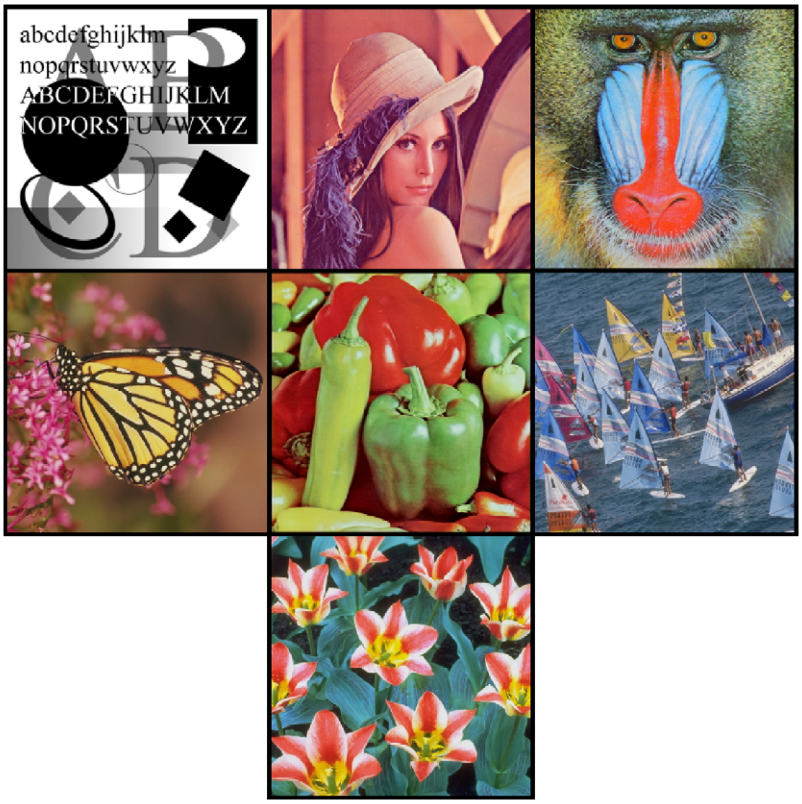

Fig. 12. Test set images.

3.2. Error measures

3.2.1. Peak signal to noise ratio

The peak signal to noise ratio (PSNR) as used in [4] is very common and based on the mean squared error (MSE).

The MSE is simply the mean of the squared differences for every channel for every pixel. Letting x_{ic} denote the value of pixel i in channel c of the original image, y_{ic} the value of pixel i in channel c of the compared image, n the number of pixels and m the number of channels, the MSE can be obtained using Eq. (8):

$$\mathrm{MSE} = \frac{1}{nm}\sum_{i=1}^{n}\sum_{c=1}^{m}(x_{ic}-y_{ic})^2 \tag{8}$$

The PSNR can be obtained from the MSE and the maximum signal value s using Eq. (9):

$$\mathrm{PSNR} = 10\,\log_{10}(s^2/\mathrm{MSE}) \tag{9}$$

The PSNR is expressed in decibels (dB) and a higher value corresponds to a lower error and thus a higher quality.

3.2.2. Structural similarity

The mean square based error measures as described in Section 3.2.1 are most widely used, but do not have a high correlation with the visual degradation in quality as noted by several authors [2,14,17,23]. In [24], Wang et al. describe and test the mean structural similarity (MSSIM) error measure. They conclude it to have a higher correlation with the visual degradation than the mean squared error (MSE), while not being a very complex measure. The structural similarity (SSIM) error measure calculates the similarity in a local window by combining differences in average and variation and correlation. It is described in detail in [24].

The SIMM measure starts with two sets of n intensity values from linked windows in the original and compared image. The averages μ_x and μ_y are calculated using Eq. (10):

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{10}$$

Next, the variances σ_x and σ_y of the two sets of intensity values are calculated using Eq. (11):

$$\sigma_x = \sqrt{\frac{1}{1-n}\sum_{i=1}^{n}(x_i-\mu_x)^2} \tag{11}$$

Finally, the correlation between the two sets of intensity values, σ_{xy}, is calculated using Eq. (12):

$$\sigma_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu_x)(y_i-\mu_y) \tag{12}$$

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Fig. 14. Sobel kernels.

The averages, variances and correlation are used to obtain the SSIM index with Eq. (13)

$$\mathrm{SSIM}_{xy} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{13}$$

the (small) constants c_1 and c_2 guarantee that the denominator cannot be equal to zero. The SSIM quality measure used in this paper has a window size of 8×8 and parameters c_1 and c_2 set to 0.0001 and 0.0009, respectively.

The local similarity measures are averaged over all possible window offsets and all channels to obtain the mean structural similarity (MSSIM), a similarity measure for the whole image. The value of the MSSIM lies between 0 and 1 and a higher value denotes a higher structural similarity and thus a higher quality.

3.2.3. Edge stability

A range of error measures has been tested in [23]. Edge stability is mentioned as being the most sensitive to the whole set of distortions and being sensitive to a blurring distortion. Since the most common distortion in super-resolution images is blurring, edge stability is chosen here as an appropriate error measure.

The edge stability error measure uses five canny edge detectors [25] with different blur deviations to obtain an ordered set of five edge maps.

The canny edge detector used starts with a Gaussian blur with the specified deviation. The horizontal and vertical Sobel kernels as shown in Fig. 14 are applied to the blurred image.

The two kernel filtered images are combined to form an edge intensity and direction map. This map is thinned and a threshold set to 0.9 times the minimum value plus 0.1 times the maximum value is applied. The edge maps are combined to a consecutive edge map by counting the maximum number of consecutive occurrences of an edge at each pixel in the ordered set of edge maps as visualized in Fig. 15. Note that all edge maps in Fig. 15 are inverted and normalised for visibility.

The resulting consecutive edge map contains numbers ranging from 0 to 5. The edge stability mean squared error (ESMSE) between the reference and compared image is

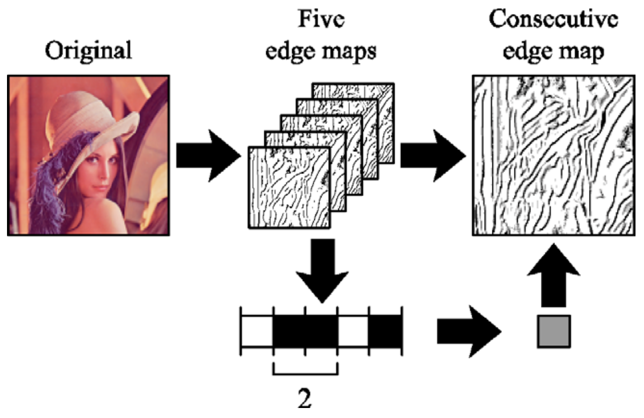

Fig. 15. Consecutive edge map.

Table 2
PSNR results for even super-resolution in dB, higher is better

	LIN	LANC	RS	NNSE	LCSR	SIA-DLIN	SIA-DLANC
128–256							
Graphic	19.67	21.29	22.77	**23.23**	21.91	20.08	22.18
Lena	29.81	31.26	**32.39**	31.93	31.93	30.47	31.68
Mandrill	23.81	24.29	**24.49**	24.29	24.41	23.89	24.33
Monarch	25.45	27.28	**28.81**	28.20	28.24	26.11	28.17
Peppers	28.75	30.27	**31.03**	30.14	30.14	29.29	30.53
Sail	25.64	26.52	**27.20**	27.12	26.98	25.92	26.82
Tulips	26.79	28.63	**30.29**	28.87	29.63	27.73	29.50
128–512							
Graphic	17.94	18.98	**20.19**	19.43	19.55	18.07	19.44
Lena	27.86	28.70	**29.57**	28.85	29.08	27.98	28.92
Mandrill	20.40	20.60	**20.71**	20.58	20.63	20.41	20.63
Monarch	23.91	25.11	**26.41**	25.22	25.90	24.08	25.63
Peppers	25.31	26.02	**26.26**	25.72	25.66	25.37	26.15
Sail	23.54	24.03	**24.63**	24.08	24.31	23.58	24.16
Tulips	25.43	26.69	**28.19**	26.14	27.56	25.67	27.24
256–512							
Graphic	22.48	24.15	26.69	**27.38**	25.77	23.25	25.61
Lena	31.89	33.13	**34.13**	33.76	33.67	32.33	33.20
Mandrill	22.63	23.24	**23.47**	23.38	23.37	22.70	23.31
Monarch	28.75	30.75	**33.22**	32.89	32.45	29.85	31.83
Peppers	28.44	**29.38**	29.07	28.66	28.41	28.56	29.18
Sail	27.20	28.61	29.67	**29.76**	29.25	27.62	29.04
Tulips	30.77	33.11	**35.53**	34.10	34.40	31.95	33.85

calculated with Eq. (14), where *ex* is the original consecutive edge map, *ey* is the compared consecutive edge map and *n* is equal to the number of edges that are detected in at least one of these two consecutive edge maps:

$$\mathrm{ESMSE} = \frac{1}{n}\sum_{i=1}^{n}(ex_i - ey_i)^2 \tag{14}$$

The edge stability (ESMSE) quality measure used in this paper uses blur deviations 1.19, 1.44, 1.68, 2.0 and 2.38. Lower

Table 3
MSSIM results for even super-resolution, higher is better

	LIN	LANC	RS	NNSE	LCSR	SIA-DLIN	SIA-DLANC
128–256							
Graphic	0.850	0.891	0.929	**0.939**	0.920	0.870	0.916
Lena	0.904	0.930	**0.940**	0.932	0.937	0.912	0.935
Mandrill	0.701	0.756	**0.779**	0.768	0.772	0.709	0.768
Monarch	0.920	0.947	**0.960**	0.953	0.956	0.930	0.955
Peppers	0.926	0.946	**0.954**	0.941	0.951	0.933	0.950
Sail	0.780	0.828	0.854	**0.854**	0.845	0.791	0.843
Tulips	0.889	0.921	**0.935**	0.912	0.932	0.903	0.930
128–512							
Graphic	0.775	0.800	**0.864**	0.798	0.854	0.784	0.823
Lena	0.778	0.805	**0.821**	0.805	0.810	0.781	0.810
Mandrill	0.459	0.502	**0.536**	0.519	0.522	0.460	0.509
Monarch	0.848	0.873	**0.896**	0.865	0.889	0.852	0.882
Peppers	0.838	0.859	**0.873**	0.840	0.864	0.840	0.864
Sail	0.586	0.633	**0.679**	0.653	0.657	0.590	0.642
Tulips	0.779	0.812	**0.843**	0.785	0.831	0.784	0.822
256–512							
Graphic	0.916	0.934	0.963	**0.969**	0.960	0.932	0.953
Lena	0.879	0.900	**0.907**	0.904	0.906	0.884	0.902
Mandrill	0.696	0.758	**0.784**	0.778	0.777	0.704	0.773
Monarch	0.940	0.957	**0.967**	0.964	0.964	0.948	0.962
Peppers	0.926	0.940	**0.945**	0.939	0.943	0.929	0.941
Sail	0.820	0.874	0.899	**0.903**	0.888	0.834	0.888
Tulips	0.923	0.952	**0.962**	0.951	0.957	0.933	0.957

Table 4
ESMSE results for even super-resolution, lower is better

	LIN	LANC	RS	NNSE	LCSR	SIADLIN	SIA-DLANC
128–256							
Graphic	1.331	1.504	1.226	1.171	1.189	1.276	1.617
Lena	1.776	1.408	**1.321**	1.331	1.485	1.769	1.377
Mandrill	2.147	1.816	**1.791**	1.795	1.921	2.125	1.860
Monarch	2.299	1.926	**1.845**	1.866	1.952	2.346	1.948
Peppers	1.807	1.391	**1.327**	1.362	1.466	1.753	1.432
Sail	1.898	1.584	1.514	1.465	1.621	1.882	1.626
Tulips	1.631	1.198	**1.146**	1.167	1.272	1.552	1.225
128–512							
Graphic	3.309	4.689	**2.998**	4.175	3.098	3.176	4.519
Lena	5.480	4.908	4.718	4.840	**4.706**	5.461	4.915
Mandrill	6.609	6.538	6.301	6.472	**6.278**	6.563	6.503
Monarch	5.448	4.850	**4.518**	4.783	4.606	5.408	4.854
Peppers	5.531	5.081	4.905	5.057	**4.864**	5.522	5.079
Sail	6.211	6.230	**5.776**	6.136	5.808	6.172	6.166
Tulips	5.994	5.664	**5.198**	5.627	5.286	5.927	5.590
256–512							
Graphic	1.059	1.367	1.085	0.981	**0.974**	1.045	1.360
Lena	1.971	1.679	**1.632**	1.645	1.728	1.964	1.686
Mandrill	2.067	1.750	**1.705**	1.723	1.813	2.097	1.777
Monarch	2.298	1.952	**1.884**	1.894	1.964	2.442	1.977
Peppers	1.942	1.575	**1.524**	1.525	1.641	1.959	1.616
Sail	1.656	1.237	1.151	**1.128**	1.282	1.613	1.277
Tulips	1.505	1.074	0.973	**0.963**	1.151	1.472	1.104

Table 5
PSNR results for odd super-resolution, higher is better

	LIN	LANC	NEDI	LAZA
128–255				
Graphic	21.57	**22.92**	21.23	20.92
Lena	31.67	**32.87**	32.45	30.68
Mandrill	26.13	**26.47**	26.16	24.97
Monarch	27.13	**28.74**	27.18	26.27
Peppers	30.47	**31.53**	30.88	29.05
Sail	27.47	**28.18**	27.67	26.63
Tulips	28.50	**30.07**	29.05	26.82
128–509				
Graphic	18.08	**19.00**	17.83	17.59
Lena	27.86	**28.66**	28.44	27.02
Mandrill	20.56	**20.76**	20.60	19.95
Monarch	23.82	**24.95**	24.03	23.06
Peppers	27.46	**28.27**	27.90	26.11
Sail	23.47	**23.95**	23.67	22.84
Tulips	25.43	**26.61**	26.00	23.87
256–511				
Graphic	22.53	**23.59**	22.41	22.24
Lena	31.81	**32.59**	32.25	31.08
Mandrill	22.81	**23.21**	22.90	22.17
Monarch	28.60	**29.94**	29.10	28.35
Peppers	32.02	**32.67**	32.30	31.14
Sail	27.14	**28.04**	27.56	26.69
Tulips	30.71	**32.16**	31.57	29.62

Table 6
MSSIM results for odd super-resolution, higher is better

	LIN	LANC	NEDI	LAZA
128–255				
Graphic	0.898	**0.915**	0.884	0.894
Lena	0.936	**0.947**	0.939	0.915
Mandrill	0.793	**0.818**	0.791	0.730
Monarch	0.944	**0.958**	0.943	0.923
Peppers	0.950	**0.959**	0.951	0.921
Sail	0.846	**0.870**	0.853	0.816
Tulips	0.926	**0.942**	0.928	0.878
128–509				
Graphic	0.780	**0.801**	0.761	0.780
Lena	0.777	**0.802**	0.788	0.749
Mandrill	0.459	**0.498**	0.463	0.390
Monarch	0.846	**0.870**	0.848	0.814
Peppers	0.845	**0.865**	0.851	0.802
Sail	0.582	**0.625**	0.596	0.542
Tulips	0.779	**0.811**	0.787	0.717
256–511				
Graphic	0.917	**0.926**	0.908	0.918
Lena	0.879	**0.889**	0.879	0.861
Mandrill	0.697	**0.732**	0.699	0.643
Monarch	0.938	**0.948**	0.941	0.930
Peppers	0.931	**0.938**	0.930	0.913
Sail	0.818	**0.852**	0.833	0.799
Tulips	0.920	**0.938**	0.926	0.894

values of the ESMSE denote higher edge stability and thus a higher quality.

4. Results

Several super-resolution algorithms are compared using PSNR, MSSIM and ESMSE error measures, which are described in Section 3.2.3. Because of the differences in super-resolution with even and odd parity, the objective results are split into these two parities. For reference linear kernel resize (LIN) and Lanczos kernel resize with radius 3 (LANC) are included in both the even and odd parity tests. A toolkit implemented for this paper was used to among others calculate objective quality measures and perform kernel super-resolution.

Table 7
ESMSE results for odd super-resolution, lower is better

	LIN	LANC	NEDI	LAZA
128–255				
Graphic	**1.032**	1.402	1.454	1.222
Lena	1.513	**1.312**	1.735	2.096
Mandrill	1.751	**1.604**	1.926	2.816
Monarch	2.035	**1.816**	2.416	2.946
Peppers	1.430	**1.225**	1.783	2.179
Sail	1.521	**1.338**	1.799	2.391
Tulips	1.273	**1.053**	1.565	2.277
128–509				
Graphic	3.258	4.602	4.552	**3.256**
Lena	5.476	**4.806**	5.365	5.661
Mandrill	6.443	6.366	**6.289**	6.999
Monarch	5.386	**4.771**	5.493	5.494
Peppers	5.339	**4.839**	5.462	5.601
Sail	6.135	6.128	**5.959**	6.642
Tulips	5.830	**5.490**	5.572	5.966
256–511				
Graphic	1.030	1.316	1.196	1.107
Lena	1.706	**1.445**	1.899	2.250
Mandrill	1.553	**1.242**	1.871	2.663
Monarch	2.074	**1.814**	2.432	2.880
Peppers	1.690	**1.433**	2.091	2.323
Sail	1.440	**1.101**	1.700	2.244
Tulips	1.402	**1.082**	1.717	2.098

Table 8
Even combined results

No.	PSNR	MSSIM	ESMSE
1	RS	RS	RS
2	NNSE	LCSR	LCSR
3	LCSR	SIADLANC	NNSE
4	SIADLANC	NNSE	LANC
5	LANC	LANC	SIADLANC
6	SIADLIN	SIADLIN	SIADLIN
7	LIN	LIN	LIN

Super-resolution results from the RS and NNSE algorithms have been kindly provided by Atkins and Staelin, respectively. Matlab code for the LCSR and SIAD algorithms has been kindly provided by Candocia and Stanco, respectively. The SIAD algorithm is performed using the toolkit mentioned earlier for resizing and the provided Matlab code for anisotropic diffusion. The SIAD algorithm is included in the test using both a linear kernel (SIADLIN) and a Lanczos kernel (SIADLANC) for initial enlargement. The NEDI algorithm has been implemented in the toolkit and is applied using a 8×8 window. The LAZA algorithm was performed using an online applet mentioned by Battiato.

PSNR results for even super-resolution are shown in Table 2. MSSIM results for even super-resolution are shown in Table 3. ESMSE results for even super-resolution are shown in Table 4. The results for odd super-resolution are shown in Tables 5–7. The best result in each row is marked with bold face.

If we sum the above results over all seven images and all three resolutions, we get a general result for each algorithm and error measure. If we order this list from best result down, we get the list shown in Table 8 for the even tests and the list shown in Table 9 for the odd tests.

In these tests, the RS algorithm performs best when tested with PSNR, MSSIM and ESMSE, but, since many algorithms require training, a bigger test set might increase performance of several algorithms.

For a subjective comparison of the results, a 80×80 pixel part of the lena image after super-resolution from 256 to 512 is

Table 9
Odd combined results

No.	PSNR	MSSIM	ESMSE
1	LANC	LANC	LANC
2	NEDI	NEDI	LIN
3	LIN	LIN	NEDI
4	LAZA	LAZA	LAZA

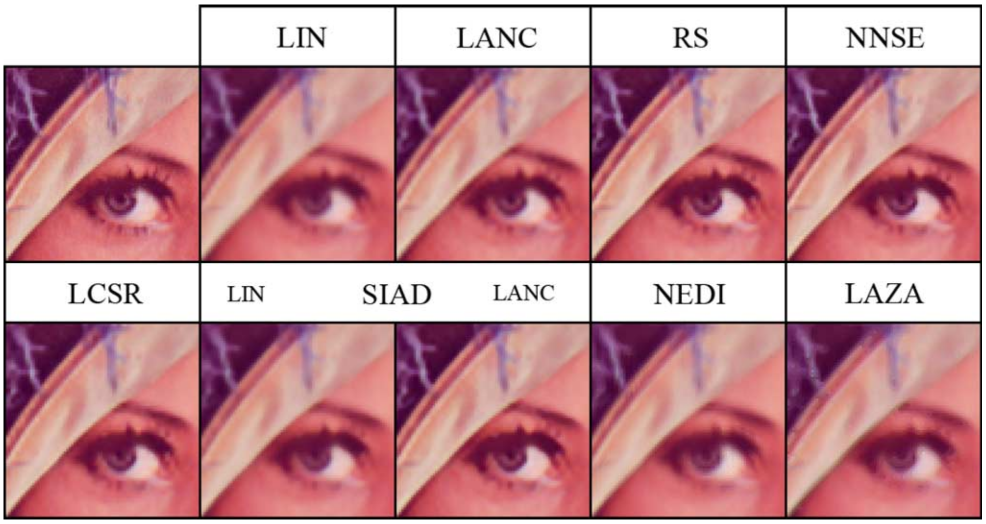

Fig. 16. Detail of lena after super-resolution from 256 by one octave.

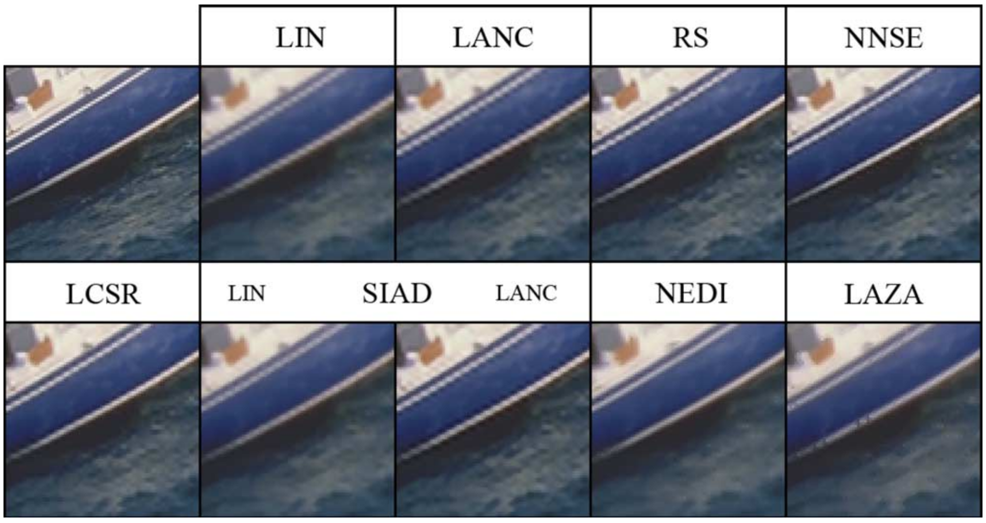

Fig. 17. Detail of sail after super-resolution from 256 by one octave.

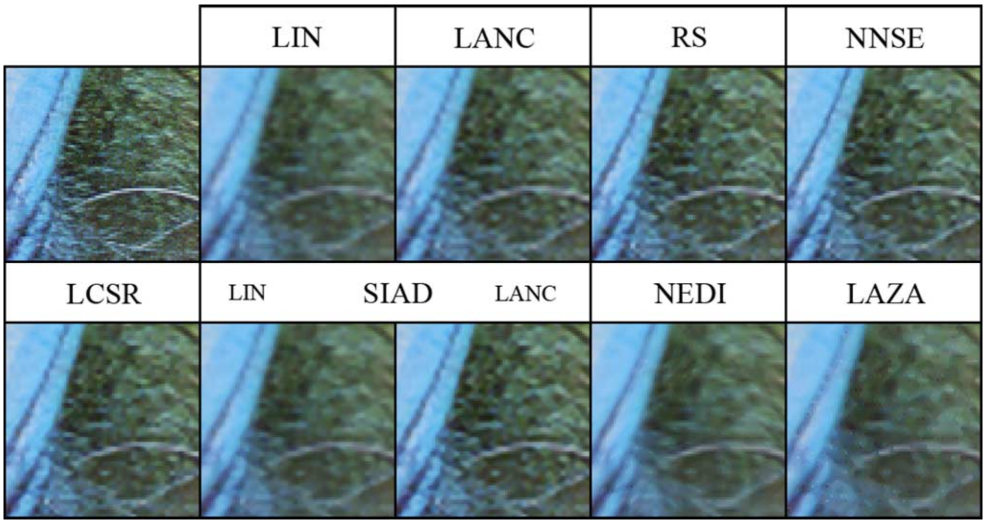

Fig. 18. Detail of mandrill after super-resolution from 256 by one octave.

shown in Fig. 16. The top left image is the original and the name of the corresponding algorithm is above the other images.

Looking at the amount blur introduced, it can be seen that RS, NNSE, LCSR and SIADLANC algorithms introduce the least amount of blur. This can for example be seen in the rim of the hat and the eye. The RS algorithm has generated a single brighter pixel as a reflection detail in the eye, while other techniques did not.

An 80×80 pixel part of the sail image enlarged from 256 to 512 is shown in Fig. 17. The diagonal line on the boat in the centre of the detail shows various levels of blocking artefacts over the enlargement. Only the NEDI and LAZA algorithms show no blocking artefacts at all at the expense of being more blurry than other techniques. The NNSE algorithm hardly shows any blocking artefacts, while belonging to the group of algorithms showing the least blurry super-resolution image.

A 80×80 pixel part of the mandrill image enlarged from 256 to 512 is shown in Fig. 18. It can be seen that all algorithms have problems generating the fine detail in the hairs of the mandrill. The amount of blur introduced is least using the RS and NNSE algorithms and slightly more in the LCSR and SIADLANC algorithms.

5. Conclusions

Looking at the amount of edge blurring in the super-resolution results first, it seems that some algorithms do a very good job at preserving the sharpness of edges; most notably the RS and NNSE algorithms, but also the LCSR and SIADLANC algorithms. It seems that this most well known issue of bilinear super-resolution can be solved in a number of different ways. When it comes to lines instead of edges, the results are less good in general. So, as one might expect, it seems that lines or double edges pose more difficulty than single edges. Since lines are rarer than edges in natural images, an improved training set might improve this for the trained algorithms.

Looking at blocking artifacts on edges, the conclusion is again that some algorithms do a very good job at reducing blocking artifacts; most notably the NNSE, NEDI and LAZA algorithms, but also the RS and LCSR algorithms. Lines again offer a bigger difficulty than edges.

After looking at blurring and blocking we come to the generation of detail. Looking at the results, this seems to be the most difficult part of the three. None of the algorithms can generate the full amount of detail present in the original image. This is very noticeable in the super-resolution results of the mandrill image, since it contains a lot of fine detail. The RS algorithm generates a single brighter pixel as reflection in the eye on the lena image, but in general, both the RS and NNSE algorithms generate the most amount of detail. The LCSR and SIADLANC algorithms are just below that.

An overview of the subjective strengths and weaknesses is presented in Table 10 using a four step scale consisting of $--$, $-$, $+$ and $++$ from low to high.

The algorithms using classification (RS and LCSR) show promising results. Classifying pixels and only applying an interpolator to the tasks it is good at, seems to be a successful way of improving overall quality. It might be an appropriate way of keeping the interpolator complexity and the overall complexity of the algorithm as low as possible, while being able to get complex results.

Table 10
Strengths and weaknesses

Algorithm	Blurring	Blocking	Detail
LIN	$--$	$-$	$--$
LANC	$-$	$--$	$-$
RS	$++$	$+$	$++$
NNSE	$++$	$++$	$++$
LCSR	$+$	$+$	$+$
SIADLIN	$--$	$-$	$-$
SIADLANC	$+$	$-$	$+$
NEDI	$--$	$++$	$-$
LAZA	$--$	$++$	$-$

Incorporating a more visually correct error measure than MSE like in the NNSE algorithm also seems to improve quality. An interpolator trained in this way cannot only obtain a more visually pleasing super-resolution image, but also one that has a good MSE quality.

Super-resolution images obtained by the NEDI algorithm are quite blurry, but do provide a visually appealing image without blocking effects. The resulting images can generally be improved by applying a sharpening filter. The ability of covariance-based enlargement to remove blocking artifacts might make it useful as an input for other methods. A neural interpolator might for example benefit from local covariance information.

Not all trained algorithms were trained using the same test set. It has been noted in several papers though that typical constructs in one image can be used as training material so to recognize similar constructs in other images. While the content of images in different training sets may differ greatly, they will probably account to similar training results, as long as both training sets consist of a fairly large set of natural images. Taking this into account and perceiving the training set as part of the super-resolution algorithm, the comparisons made above are legitimate. This of course does not mean that a difference in training set could not influence the quality of the super-resolution algorithm.

When breaking down the algorithms reviewed in this paper into elemental parts like a linear associative memory part or a clustering part, it seems that a lot of the algorithms can be combined. It seems possible for example to have a belief propagation part using both the neural networks from NNSE and the patch lookup from RS as an information source. In this way it might be possible to further improve the state of the art algorithms described in this paper.

When looking to the general results in the tests, it seems that handling blurring and blocking for a two by two enlargement is becoming more and more feasible, but that the generation of detail is still a difficult point.

References

[1] C.B. Atkins, C.A. Bouman, J.P. Allebach, Optimal image scaling using pixel classification, Proceedings of International Conference on Image Processing, 2001, pp. 864–867.

[2] C. Staelin, D. Greig, M. Fischer, R. Maurer, Neural network image scaling using spatial errors, HP Laboratories Israel, October 2003.

[3] D.D. Muresan, T.W. Parks, Adaptive, optimal-recovery image interpolation, IEEE International Conference on Acoustics, Speech, and Signal Processing 3 (2001) 1949–1952.

[4] F.M. Candocia, J.C. Principe, Superresolution of images based on local correlations, IEEE Transactions on Neural Networks 10 (2) (1999) 372–380.

[5] S. Battiato, G. Gallo, F. Stanco, A locally-adaptive zooming algorithm for digital images, Image Vision and Computing Journal 20 (11) (2002) 805–812.

[6] X. Li, M.T. Orchard, New edge-directed interpolation, IEEE Transactions on Image Processing 10 (10) (2001) 1521–1527.

[7] X. Xu, L. Ma, S.H. Soon, C.K.Y. Tony, Image interpolation based on the wavelet and fractal, International Journal of Information Technology 7 (2) (2001).

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

[9] S. Battiato, G. Gallo, F. Stanco, Smart interpolation by anisotropic diffusion, Proceedings of 12th International Conference on Image Analysis and Processing, 2003, pp. 572–577.

[10] D.D. Muresan, T.W. Parks, Adaptively quadratic (AQua) image interpolation, IEEE Transactions on Image Processing 13 (5) (2004) 690–698.

[11] D. Su, P. Willis, Image interpolation by pixel level data-dependent triangulation, Computer Graphics Forum 23 (2) (2004).

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

[13] W.T. Freeman, E.C. Pasztor, Markov networks for super-resolution, Proceedings of 34th Annual Conference on Information Sciences and Systems, March 2000.

[14] M.F. Tappen, B.C. Russell, W.T. Freeman, Exploiting the Sparse Derivative Prior for Super-Resolution and Image Demosaicing, 2003.

[15] C.B. Atkins, C.A. Bouman, J.P. Allebach, Tree-based resolution synthesis, Proceedings of IEEE ICIP (1999) 405–410.

[16] S. Battiato, G. Gallo, M. Mancuso, G. Messina, F. Stanco, Analysis and characterization of super-resolution reconstruction methods, Proceedings of SPIE Electronic Imaging 2003, January 2003.

[17] D.D. Muresan, T.W. Parks, Image interpolation using adaptive linear functions and domains, IEEE 2001 Western New York Image Processing Workshop, 2001.

[18] D.D. Muresan, T.W. Parks, Optimal recovery approach to image interpolation, Proceedings of IEEE ICIP, 2001.

[19] D.D. Muresan, T.W. Parks, Prediction of image detail, Proceedings of IEEE ICIP, September 2000.

[20] K. Kinebuchi, D.D. Muresan, T.W. Parks, Image interpolation using wavelet-based hidden markov trees, IEEE ICASSP, 2001.

[21] N.A. Dodgson, Quadratic interpolation for image resampling, IEEE Transactions on Image Processing 6 (9) (1997) 1322–1326.

[22] M.J. Black, G. Sapiro, D.H. Marimont, D. Heeger, Robust anisotropic diffusion, IEEE Transactions on Image Processing 7 (3) (1998) 421–432.

[23] İ. Avcıbaş, B. Sankur, K. Sayood, Statistical evaluation of image quality measures, Journal of Electronic Imaging 11 (2) (2002) 206–223.

[24] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Transactions on Image Processing 13 (4) (2004) 600–612.

[25] J.F. Canny. A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6) 1986, pp. 679–698.

[26] X. Yu, B.S. Morse, T.W. Sederberg, Image reconstruction using data-dependent triangulation, IEEE Computer Graphics and Application 21 (3) (2001) 62–68.

[27] K. Jensen, D. Anastassiou, Subpixel edge localization and the interpolation of still images, IEEE Transactions on Image Processing 4 (3) (1995) 285–295.

[28] http://www.benvista.com.

[29] http://www.kneson.com.

[30] http://www.ddisoftware.com.

[31] http://www.dmmd.net.

[32] http://www.extensis.com.