Overlapping DCT for image compression

R.J. Henery
Dept. Statistics and Modelling Science
Strathclyde University
Richmond Street
Glasgow G1 1XH

Monday, 1st July 1996

Summary

Most current research in image compression (wavelets, local trigonometric bases, JPEG, etc.) centres on adaptive coding of the grey levels of the image. These methods may be described as using band-limiting procedures with a band-limit that depends on the local properties of the image. This paper looks at the non-adaptive use of Discrete Cosine Transforms in blocks, with uniform band-limits over the whole image.

The relationship between compressed Cosine Transforms and interpolation is used to establish a new type of lapped Cosine Transform, allowing high compression of images without the well-known "quilting" effect of JPEG-style compression. The method is fast and easily programmed, and achieves results comparable with the best wavelet compression methods.

The method is illustrated on a well-known image (Lena 512). For this particular image at least, it appears that non-adaptive bandlimiting does at least as well as adaptive methods in the very high limit, with no visible signs of quilting at a compression ratio of 250 to 1.

Keywords
DCT, Wavelets, Image compression, Interpolation, Hierarchical

reconstruction, Multiple Resolution, Reproducing Kernel Hilbert Spaces.

1 Introduction

The aim of the JPEG group was to be "at or near state of the art" for compression of images at the "excellent quality" end of the compression range (Wallace (1991)). Several waveleteers believe that JPEG is no longer "state of the art". For instance Hilton, Jawerth & Sengupta (1994) say that "The reconstruction quality of wavelet compressed images has already moved well beyond the capabilities of JPEG, which is the current international standard for image compression". Further evidence that wavelet methods do well is evidenced by the FBI wavelet image compression standard, which provides good quality images from compression rates of around 20:1 (Bradley, Brislawn & Hopper (1994)). Such claims are usually based on the quilting effect of JPEG at high compression and low image quality. We propose an overlapped version of the Discrete Cosine Transform Methods that is especially useful **in the high compression region**. The principal idea is to use the interpolation properties of the Discrete Cosine Transform (DCT) to establish an image compression method capable of very high compression ratios yet avoiding the well-known quilting of the JPEG system. The compression stage is very close to JPEG, and consists of three steps that may be described as blocking, filtering and downsampling. Further compression can be achieved by coding the results, but this is not our main interest. The main difference with JPEG or with wavelet methods, is that the method relies on bandlimiting the image in blocks **using the same bandwidth in each and every block**. Thus the method is not spatially adaptive in any way. In a further departure from the usual JPEG, at the reconstruction stage the blocks are overlapped giving rise to a continuous approximation to the original signal. The results are comparable to the best wavelet methods at extremely high compression ratios (typically at compression ratios of around 250).

We suggest two reasons for considering such high compressions:-

1. It is quite common on the Internet to show a small image, or a whole series of small images, alongside some descriptive text, with the instruction to "click here to view a full-size version". These small images, or icons as we will call them, are typically derived from the full

size image by subsampling every 4'th, 8'th or 16'th pixel, giving rise to compression ratios in the range 16-256. The quality of these icons can often be improved very greatly by "blurring" the image before subsampling, following the general principles of wavelet theory. In short, the recommendation is to apply a smoothing filter before subsampling. A related analogy in wavelet theory is the construction of low resolution maps from a high resolution map, as described in the book by Meyer (1993), for example.

2. We can use a highly compressed image as a basis for a predictive **lossless** compression scheme, provided that the reconstruction from the compressed image is a reasonably good approximation to the original. Entropy considerations enable us to predict what level of **lossless** compression using predictive compression starting from any given **lossy** high-compression scheme.

1.1 Lena Image

To illustrate the procedures, we will use the well-known Lena image in the 512*512 8 bits per pixel form and the file: `GreySet2/lena`, which is "from the Y component of a YUV decomposition". This file is available from the Waterloo ftp site (maintained by John Kominek)
`http://links.uwaterloo.ca/pub/BragZone/`.

Rather than show the full 512*512 image, we show two icons of size 64*64 in Figure 1. The left icon is obtained by subsampling every 8th pixel of the original, following the usual practice on the Internet. The right icon is produced by first smoothing the original, and then subsampling every 8'th pixel. The interested reader can mimic these processes by using the `xv` package. The left icon is produced by halving the image size three times: the right icon is "blurred" (with a 5*5 blur) before halving three times. Subjectively, the right icon is a better representation of the original. Objectively, it is clear that the left icon is much noisier. It is also clear that some vertical or horizontal lines may be absent completely if smoothing is not involved before the subsampling stage.

For explanatory purposes, it is convenient to use the language of signal processing, although the main application considered is image processing. Thus we start by considering a sequence of observations in time such as might arise from uniform sampling of a continuous signal. The discrete time signal must be compressed as much as possible, without sacrificing accuracy too much. Our main concern here is to obtain a highly compressed signal which, when uncompressed and reconstructed, leads to a continuous approximation to the original (pre-sampled continuous) signal.

1.2 Blocking

The discrete-time signal is divided into blocks of N points (in JPEG N=8 normally). The discrete cosine transform is applied to each block separately, giving rise to N cosine transform coefficients corresponding to frequencies in the range 0,..,(N-1). The zero-order frequency is referred to as the DC component.

1.3 Filtering

Compression is achieved by strict bandlimiting within each block, setting to zero all frequencies above the first M components of the spectrum, i.e. setting to zero the coefficients for frequencies M,...,(N-1). The inverse DCT of this set of N DCT coefficients (with the last $N - M$ coefficients zero) forms a filtered approximation to the original signal. As all the high frequency components have been removed, the resulting approximation in each block is smoother than the original: how much smoother depends on the magnitudes of omitted components. However, as as is well known, there will be discontinuities between the blocks, and these discontinuities spoil the overall smooth picture.

1.4 Downsampling

Taking only the first M components of the spectrum within each block and performing the inverse DCT (of length M) we obtain M points in time: this is the downsampling step. Two points are noteworthy: (i) the M downsampled points are further apart than the original points, in the ratio N/M, so the resolution of the image is lower by this ratio; (ii) we have achieved a compression of N/M in the amount of data to be stored.

1.5 Reconstruction

Reconstruction starts from these M time-related values. The two steps involved are: (i) discrete cosine transform and (ii) upsampling.

DCT. The compressed signal can, of course, be unpacked by taking the DCT of successive blocks of M points. **Upsampling**. All that is necessary is to adjoin (N-M) zeros to the DCT spectrum and take the inverse DCT of the result to obtain the upsampled approximation to the original signal.

1.6 Upsampling via overlapping blocks

The main problem with the upsampling scheme just outlined is the discontinuities between the blocks. Yet these discontinuities can be avoided very simply by using a simple interpolation scheme based on overlapping blocks of the compressed signal. The scheme is obviously very close in motivation to the Lapped Transforms of Malvar (1990), Coifman & Meyer (1991) and Jawerth, Liu & Sweldens (1995), the distinction being that these authors use overlapping transforms on the original data as a prefilter stage. One advantage of overlapping on the compressed data is the reduction in overheads as the compressed data will be very much smaller than the original in our applications. There are conceptual advantages also, as the present method makes clear the close connection between upsampling and interpolation.

2 Discrete Cosine Transform DCT

Let f_v be the observed values of the signal at N uniformly sampled time points, $v = 0, 1, ..., (N-1)$. As an aid to remembering the notation, we use Roman letters t,u,v,... to denote time-related variables and the Greek letter ω to denote the frequency variable.

The discrete cosine transform DCT of f_v is given by

$$F_\omega = \frac{4}{N} C_\omega \sum_{v=0}^{N-1} f_v \cos \frac{(2v+1)\omega\pi}{2N}; \quad \omega = 0, 1, \ldots, (N-1) \tag{1}$$

where $C_\omega = 1/\sqrt{2}$ for $\omega = 0$ and $C_\omega = 1$ for $\omega = 1, \ldots, (N-1)$.

2.1 Inverse DCT as interpolation formula

The inverse discrete cosine transform IDCT, which is of course exact, is

$$f_v = \frac{1}{2} C_\omega \sum_{\omega=0}^{N-1} F_\omega \cos \frac{(2v+1)\omega\pi}{2N}; \quad v = 0, 1, \ldots, (N-1). \tag{2}$$

Observe that we can use the formula 2 to define an interpolated signal for all time in the interval $0 < t < 1$ by setting $t = \frac{(2v+1)}{2N}$. We adopt the notation $f(t)$ for the result, to emphasize that $f(t)$ is a continuous function of time:

$$f(t) \;=\; \frac{1}{2}C_\omega \sum_{\omega=0}^{N-1} F_\omega \cos t\omega\pi; \quad 0 < t < 1. \tag{3}$$

Clearly $f(t) = f_v$ at the sampling points $t = \frac{(2v+1)}{2N}$, so that $f(t)$ is an exact interpolation formula. Note that the first sampling point occurs at $t = 1/2N$ but neighbouring sampling points are separated by an interval of $1/N$. This is the Shannon sampling theorem for discrete cosine transforms. In essence it says that we can recreate the complete function $f(t)$ over the whole interval $(0,1)$ by specifying the values at the N time points $t = \frac{(2v+1)}{2N}$, for $v = 0, \ldots, (N-1)$. The particular feature of interest here is the explicit procedure for obtaining the sampling points that enable us to reconstruct the given function exactly knowing only the values of the function at the sampled points. Such properties are also held by wavelet approximations (Daubechies 1992) who notes the connection between Shannon's sampling theorem and Reproducing Kernel Hilbert Spaces. However, wavelet approximations do not generally allow of a simple representation in terms of a set of uniformly sampled data points.

2.2 Bandwidth limited IDCT

The essence of compression is reducing the number of parameters that must be stored to reproduce the given function. One way of doing this is to store only the first M transform coefficients $F_\omega, \quad \omega = 0, 1, \ldots, (M-1)$. For approximately bandlimited functions, this will give a good approximation to the original function. Thus we may approximate f(t) by truncating the summation in equation (3) after the first M terms giving

$$\hat{f}(t) \;=\; \frac{1}{2}C_\omega \sum_{\omega=0}^{M-1} F_\omega \cos t\omega\pi; \quad 0 < t < 1. \tag{4}$$

the only change compared to 3 being in the number of terms included in the summation. However, instead of storing the M transform coefficients F_ω, we can store M function values in the time domain, these function values being the values of $\hat{f}(t)$ at the sampling points $t = \frac{(2v+1)}{2M}$, for $v = 0, \ldots, (M-1)$, and, conceptually at least, this is easier to understand than the corresponding transform coefficients.

Clearly this approximate function $\hat{f}(t)$ is the inverse transform of the first M frequency components F_ω. Of course, we have achieved a compression of N/M by this means, but the resulting approximation no longer passes through the original time points f_u.

3 Compression

Our main concern is the compression of images in the situation where the reconstructed image ("icon") will be displayed at much lower resolution than the original.

The three main elements of image compression are (i) thresholding or sampling; (ii) quantization, and (iii) lossless coding of the result.

3.1 Thresholding or Sampling

The aim in thresholding or sampling is to use the redundancy in the signal to reduce the number of function values (or the number of transformed function values) that must be stored to give an adequate approximation to the signal. Wavelet compression claims to take advantage of the localization properties of the wavelet transforms through which information tends to be concentrated in a small number of wavelet coefficients. When the original signal is bandlimited however (and some a priori information is necessary to establish this) the simplest way to compress is to throw away frequency components higher than the bandlimiting frequency M and sample at the Nyquist rate $1/M$. This is the approach adopted here. If signals are only approximately bandlimited, the error is measured by the magnitude of the signal spectrum above M. In common with most theoretical papers on compression we judge the magnitude of the error by the root mean square error σ.

3.2 Quantization

Once it is accepted that compression involves some degree of approximation, it is readily understood that numerical accuracy is not so important when it comes to storing function values or coefficients. If q is an agreed quantization level, with $q < \sigma$, function values f_u can be discretized by storing the integer value of f_u/q. This quantization step introduces an additional error of approximately $q/2\sqrt{3}$ if f_u is a continuous random variable with range much greater than q.

3.3 Lossless coding

There are many ways to code discretized signals. Generally some form of pre-processing is required to take advantage of correlations within the data. Transform methods generally do not use pre-processing, relying on the approximate independence of the transform coefficients. However, at the very least, there is usually considerable dependence within the function values or transform coefficients. We code the differenced data rather than the original data. This is appropriate for non-stationary signals only. In the context of our compression method, coding is applied to the compressed signal which is very smooth and therefore highly correlated, so some form of differencing is very beneficial. With image data, JPEG gives several options for differencing data: some experimentation is necessary to determine the optimal type of differencing.

In the context of wavelet methods, considerable improvements in compression result from using the fact when one wavelet coefficient is small, there is an increased likelihood of higher level wavelet coefficients in that locality also being small (Shapiro 1993).

3.4 Compression for Blocked DCT (a la JPEG)

Standard JPEG starts by dividing the image into square blocks (usually of size $N = 8$). Each block is treated on its own. The forward DCT coefficients F_ω are compared to a quantization table Q_ω, and a coefficient is retained if $F_\omega > 0.5 \times Q_\omega$. As a result, the highest retained frequency varies from block to block, i.e. in principle the bandwidth retained has any value in the range $0, \ldots, (N-1)$ (the DC component is always retained). This also leads to a differing compression in each block, and while this makes good sense in terms of retaining more detail when important, it is inconvenient when combining the compressed coefficients to form a single compressed signal in the time domain as the effective sampling interval varies from one block to the next. To simplify matters therefore, we use the same bandlimit in every block. The principal advantage is that the retained coefficients can be stored as an image instead of a set of coefficients in transformed space.

3.5 Icon Construction for Bandlimited Blocked DCT

The first stage proceeds as in JPEG, but usually we take a block size much greater than 8: $N = 32$ in the Lena image example in Figure 4. Compression is achieved by bandlimiting every block to the same low frequency band, for example limiting the frequencies to the lowest $M = 4$ frequencies. Taking the inverse DCT of the $M = 4$ retained frequencies, we arrive at a set of $M = 4$ function values in the time domain for each block. The compressed function values for the blocks are put together in sequence to provide a compressed version ("icon") of the original full dataset.

Figure 2 illustrates the basic method. Starting from $N = 8$ points sampled in the time domain as in Figure 2(a), the DCT is found as in Figure 2(b). After bandlimiting, the spectrum is reduced from $N = 8$ to the lowest $M = 4$ values, shown in Figure 2(c). The inverse DCT of the bandlimited spectrum is shown in Figure 2(d). Also shown in Figure 2(d) is the interpolated function for the compressed data. In practice, the simplest method for finding the reconstruction for the original function is to start from the bandlimited spectrum, adjoin a very large number of zeros, and then take the inverse DCT.

3.6 High Resolution Image from Icon

To reconstruct an approximate high resolution image starting from the low resolution icon, we must interpolate N points from each block of M of the icon. There are two obvious ways of doing this, and they are illustrated in Figure 3. Imagine that the original data consisted of a pure sine wave signal with additive white noise. After smoothing and downsampling, the compressed data will look something like the discrete (dotted) points in Figure 3(a), in fact a simple sine curve sampled 8 times in each period. The phase and frequency of this sine curve were chosen to give maximum discontinuities across non-overlapping blocks (basically so that the mean in one block is as different as possible from the mean in the next block). The dotted points represent the low resolution function that must be interpolated to produce what we hope will be a good approximation to the original function.

3.7 Non-overlapping blocks

Figure 3(b) shows the reconstruction based on **four non-overlapping** blocks each of size $M = 4$. Note the discontinuities between the blocks.

3.8 Overlapping blocks

When reconstructing an approximation to the original data from this compressed dataset, we again use blocks of size $M = 4$ in the compressed signal. However neighbouring blocks have one point in common. Thus the first block takes points 1,2,3,4 from the compressed data, the next block takes points 4,5,6,7, the next again takes points 7,8,9,10, etc. etc.. Consequently, the reconstructed approximation values are strictly interpolated values from the compressed points, that is, their time location is strictly within the range of times of a block.

Figure 3(c) shows the reconstruction based on **five overlapping blocks** also of size $M = 4$: neighbouring blocks share one point, and, as a result, the interpolated values in neighbouring blocks are guaranteed to meet at that point.

4 Image Compression

The overlapping procedure carries over directly to 2-dimensional data such as images, both dimensions being bandlimited (in blocks). With images the same bandlimit will normally be used in both dimensions. In general, some experimentation will be needed to establish the parameters involved: width of the blocks N; the bandlimit M; the quantization level q; and the form of coding.

4.1 Lena Image

For definiteness, we will describe the procedure we adopted, after a little experimentation, for a particular image, namely the well-known Lena image in the 512*512 8 bits per pixel form. As there are several versions of Lena around, we give the source: `http://links.uwaterloo.ca/` and the file: `GreySet2/lena`, which is "from the Y component of a YUV decomposition". We also used the same Waterloo source for the comparative results of other authors on this image: for details of these, please refer to the Waterloo ftp site (maintained by John Kominek). Figure 4 shows a detail from the Lena image, together with several transform coded versions at about the same compression ratios. The wavelet version Figure 4(b) was based on Daubechies Extremal Phase with 8 filter coefficients, using Universal Thresholding as detailed in Nason & Silverman (1994). The Fourier Transform version Figure 4(c) is based on the FFT of the whole image, retaining only the bottom 12.5% of the spectrum.

4.2 Icon Construction

The image was divided into 64 blocks each of size 32×32, i.e. $N = 32$. The DCT of each block was found and then bandlimited to the lowest 8×8 block, i.e. $M = 8$. The compression achieved by this step is clearly 16.

Compression by Quantization. As the signal-to-noise ratio is rather high, the compressed image was further compressed by quantizing, using a quantum level approximately equal to the rmse error. In this case this means quantizing with a step size of 16, thereby reducing the number of bits retained by a factor of two. To illustrate what happens when a 512×512 image is reduced in this way to a 64×64 image, see Figure 4(d) which shows a detail of this reduced image. Clearly much information has been lost, as you would expect with a compression ratio of 128.

Compression by differential coding. When the differenced pixel intensities from the quantized image Figure 4(d) are coded using a variable length coding, a further compression of nearly 2 is obtained. In fact the overall compression achieved for Figure 4(d) is 244.

Reconstruction by non-overlapping blocks.

Although the pixel size is much too great in Figure 4, interpolation improves the visual quality: Figures 4(e) and (f) show interpolations from Figure 4(d) based on non-overlapped and overlapped DCT's respectively. The standard JPEG-style reconstruction from the compressed data, using non-overlapping blocks, is shown in Figure 4(e). The blocking effect is clearly evident.

Reconstruction by overlapping blocks. Using overlapping blocks, the reconstruction from the compressed data gives the image shown in Figure 4(f). There is no noticeable blocking effect (although clearly the quality of the image is very poor).

4.3 Results

The results of several runs of the algorithm with various settings of the parameters are given in Table 1. The run giving rise to the images of Figure 4 is given by the last row of this table.

Method	quantum	bandwidth	block	CR	PSNR	Time (secs)
DCT	16	4	32	243.9	24.74	5.9
DCT overlap	16	4	32	243.9	24.95	7.9
FFT	16	32	512	228.4	25.01	11.1
Wavelet	constant	adaptive	512	101***q**	24.70	12.6

Table 1: Compression ratios and PSNR's for some band-limited (quantized and coded) images from Lena512. The wavelet compression is approximate, with $q \approx 2.5$.

4.4 Comparison with wavelet compression methods

For comparison, a "standard" wavelet technique was used to give a compressed image of approximately the same compression ratio. The wavelets used were the Daubechies Extremal Phase with 8 filter coefficients. The compression method was Universal Thresholding as detailed in Nason & Silverman (1994). The choices of wavelet type, filter length and method of compression were arbitrary: they are the default choices in the `wavethresh` software provided by Nason & Silverman. Figure 4(f) shows the resulting image which represents a compression of 101: further compression of the wavelet coefficients would be possible, say by a factor $q \approx 2.5$, using tree methods for example.

For comparative purposes, we calculated the root mean square error $rmse$ and Power Signal-to-Noise Ratio (PSNR)

$$PSNR \;=\; 20\log_{10}(\frac{255}{rmse}).$$

For the final images in Figures 4(b,c,e,f) the PSNR figures were 24.70, 25.01, 24.74 and 24.95 respectively, so that in a mean square sense the images are very similar in quality. There is no doubt however that the blocking effect of JPEG is distracting to the eye. And although the wavelet method does capture some local details better (compare the eyes and eyebrows in Figures 4(b,e,f)), there are some artefacts in the wavelet image that could well cause difficulty. These are similar in nature to the Gibbs phenomena seen in the Fourier version Figure 4(c). Two independent judges were asked to rank the compressed images in order of preference: both put the DCT image as better, with the `codetree` image not far behind. The quality of the images was agreed to be very poor.

5 Lossless Compression with the aid of Lossy Compression

Suppose we have a lossy compression method $\mathcal{Y}$, with compression ratio CR, mean square error σ^2, and $PSNR = 20\log_{10}(\frac{255}{\sigma})$. Assuming σ is large compared to unit difference in grey levels, the entropy of the errors ϵ is

$$H(\epsilon) \geq 0.5\log_2(2\pi\sigma^2 e)$$

measured in bits. Using the definition of $PSNR$, we obtain an inequality for the entropy of the errors ϵ:

$$H(\epsilon) \geq 10.04145 - \frac{PSNR}{6.0206}.$$

Assuming that the original image has 8 bits per pixel, lossless compression of the full image needs $H(\epsilon)$ bits to encode the errors and $8/CR$ bits for the compressed image. The overall lossless compression ratio cr for the full image is therefore, using inequality 5:

$$cr = \frac{8}{H(\epsilon) + 8/CR}$$

$$cr \geq \frac{1}{1.2552 - PSNR/48.16 + 1/CR}.$$

The significance of this inequality is that it relates lossless compression to lossy compression: maximum performance in one regime is limited by maximum performance in the other.

Taking the Lena image as example, with a lossy compression ratio of $CR = 250$ and with $PSNR = 26.0$, a lossless compression of 1.39 should be possible. In fact, Said & Pearlman's (1993) `codetree SAPA` method almost achieves this limit, while Said & Pearlman's (1993) `codetree SAPB` method achieves a lossless compression $cr = 2.0$.

6 Final Remarks

The contrast between the two types, adaptive and non-adaptive, gives some idea of the advantage of adaptivity.

There is no claim that the results are the best achievable either within this software or in wavelets generally. The timings in Table 1 given for the various methods had only two things in common: they were on the same computer (Sun workstation) in the same environment (within Splus). The Splus environment is not ideal for comparing the speeds of algorithms.

Jawerth, Liu & Sweldens (1995) also deal with Smooth Local trigonometric bases. They publish results for two variants of Smooth Local trigonometric bases (biorthogonal and equal parity), and also for JPEG. The PSNR figures for these methods do not seem to differ enough to justify preferring any one method uniformly above any other. In general, the most efficient compression techniques, as judged by PSNR vs. CR performance, are probably the wavelet-tree combination of Shapiro (1993) or the spatial orientation tree of Said & Pearlman (1993). However, Hilton, Jawerth & Sengupta (1994) maintain that such methods are much slower than pure wavelet methods, and state that the bottleneck in compressing video images is in the coding/decoding. The transform methods discussed in this paper (DCT and FFT) are at least as fast as, if not faster than, wavelet methods, and the compression methods chosen are also very fast (the times to compress and uncompress are much less than the times to transform the data). If speed of transmitting images is a major issue, as well as quality of image, a compromise between efficient compression and speed is necessary.

References

Bradley, J., Brislawn, C., and Hopper, T. (1994). The FBI wavelet/scalar quantization fingerprint image compression standard. Technical Report LA-UR-94-1409, Los Alamos National Laboratory.

Daubechies, I. (1992). *Ten Lectures on Wavelets.* SIAM, Philadelphia, USA.

Hilton, M., Jawerth, B., and Sengupta, A. (1994). Compressing still and moving images with wavelets. *Multimedia Systems*, 2(3):to appear.

Jawerth, B., Liu, Y., and Sweldens, W. (1995). Signal compression with smooth local trigonometric bases. Technical report, University of South Carolina.

Meyer, Y. (1993). *Wavelets: Algorithms and Applications.* SIAM, Philadelphia, USA.

Said, A. and Pearlman, W. (1993). Image compression using the spatial compression tree. In *Proceedings IEEE Int. Symp. Circuits and Syst.*, pages 279–292, Chicago, IL. IEEE.

Shapiro, J. (1993). Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions of Signal Processing*, 41(12):3445–3462.

Wallace, G. (1991). The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44.

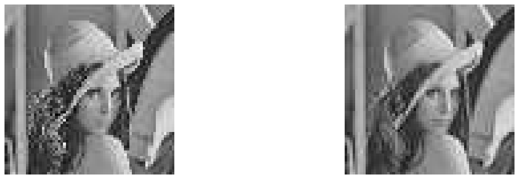

Figure 1: Left: icon produced by sampling every 8'th pixel. Right: icon produced by first smoothing (using a bandlimited DCT) and then subsampling.

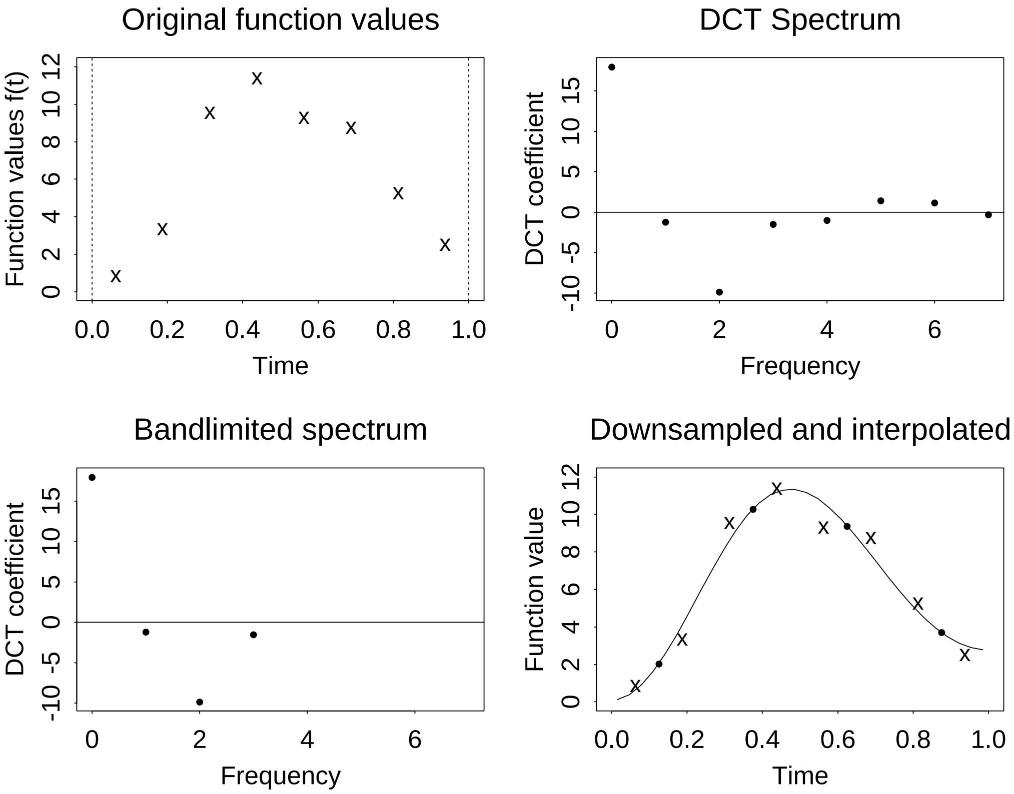

Figure 2: Illustrating the use of blocked DCT's to arrive at a smoothed and subsampled image. Bottom right: crosses represent the original data and the dots represent the smoothed and downsampled values.

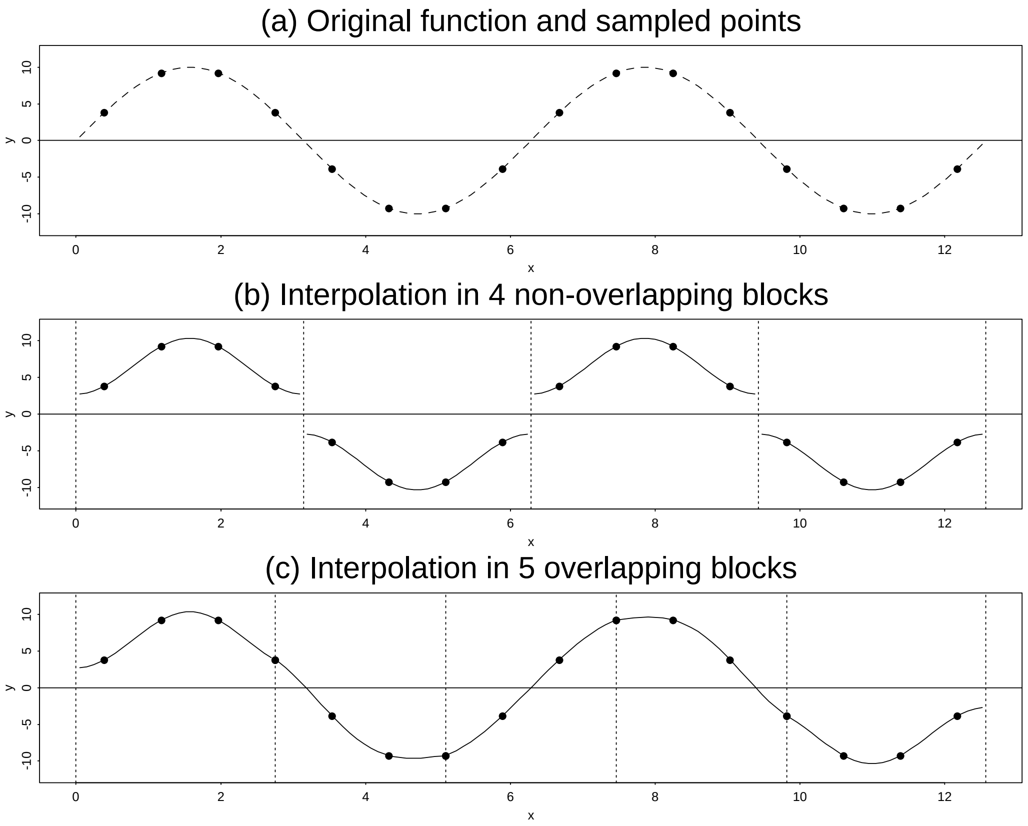

Figure 3: The use of overlapping and non-overlapping blocks for reconstruction of a signal. Figure 3(a) shows the original data: the original function is the dashed curve; the sampling points are shown dotted. Figure 3(b) shows the interpolated function based on non-overlapping blocks, each containing four points. Figure 3 shows the interpolated function based on overlapping blocks of four points, with neighbouring blocks having one point in common.

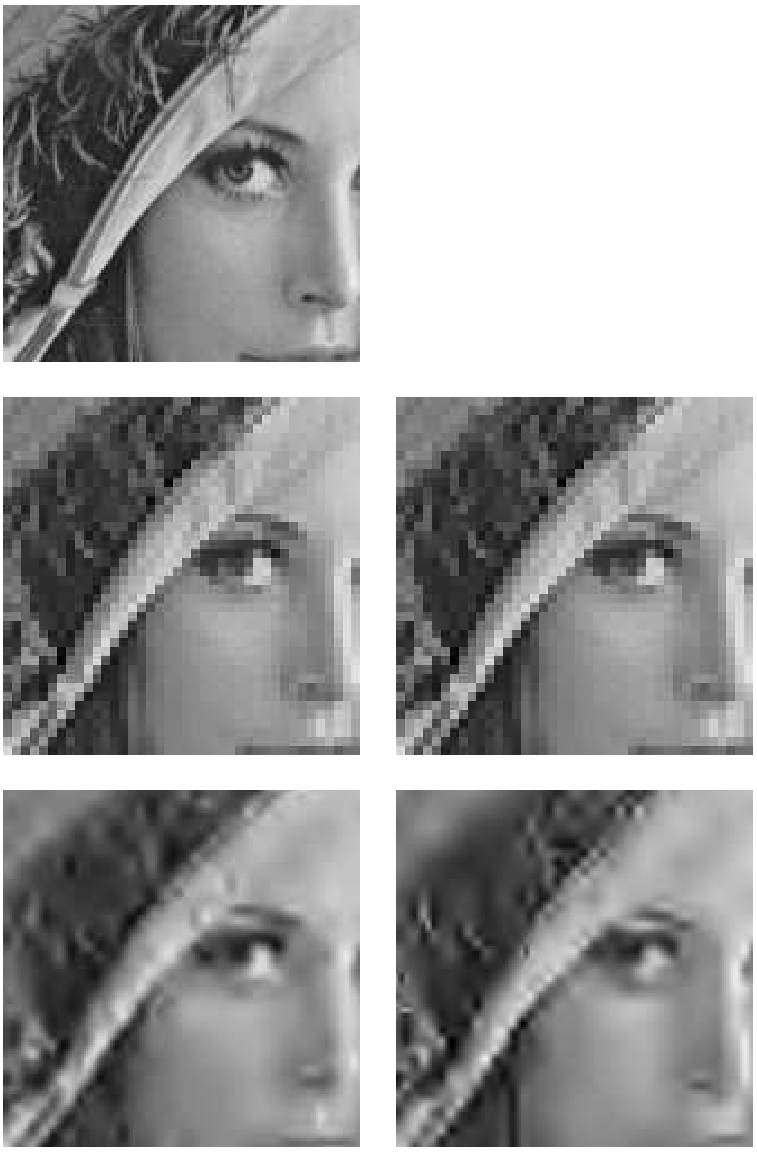

Figure 4: Detail from the Lena 512*512 image. Top left: Original. Middle: downsampled *4 (left) and compressed to 1048 bytes by `codetree` (right). Bottom: two reconstructed images from compressed files of 1048 bytes: left: interpolated from compressed 128*128 icon. Bottom right: directly compressed to 1048 bytes via `codetree` (right).