JPEG 2000 and Region of Interest Coding

Andrew P. Bradley, *Member, IEEE* {A.Bradley@unsw.edu.au}†

Fred W. M. Stentiford {Fred.Stentiford@bt.com}‡

Abstract-- **This paper details work undertaken on the application of JPEG 2000, the recent ISO/ITU-T image compression standard based on wavelet technology, to region of interest (ROI) coding. The paper briefly outlines the JPEG 2000 encoding algorithm and explains how the packet structure of the JPEG 2000 bit-stream enables an encoded image to be decoded in a variety of ways dependent upon the application. The three methods by which ROI coding can be achieved in JPEG 2000 (tiling; coefficient scaling; and code-block selection) are then outlined and their relative performance empirically investigated. The experimental results show that there are a number of parameters that control the effectiveness of ROI coding, the most important being the size and number of regions of interest, code-block size, and target bit rate. Finally, some initial results are presented on the application of ROI coding to face images.**

Index terms-- **Image and Video Coding, JPEG 2000, Wavelets.**

I. INTRODUCTION

JPEG 2000 (JP2K) is the emerging image compression standard developed jointly by the ISO/ITU-T (International Organisation for Standardisation / International Telecommunications Union), to complement the current JPEG standard [1] by providing improved compression performance and new functionalities [2]. JPEG 2000 Part I, the core coding algorithm, became an international standard in December 2000 [7] and provides, in a single bit-stream, a wide array of functionalities, such as: progressive transmission by resolution, quality, component, or location; random access; loss-less to lossy compression; and error tolerance. The functionality that is investigated in detail in this paper is the ability to encode a region of interest (ROI) with more detail than the image background [4],[5].

The JP2K algorithm is based upon embedded block coding with optimised truncation (EBCOT) [6]. JP2K has been designed to offer compression performance as good as, or better than, conventional JPEG. Reported results comparing JP2K to conventional JPEG [3] indicate approximately a 2dB improvement in image quality (measured with PSNR) at the same bit rate, or alternatively, a 20-30% improvement in compression for the same quality. The superiority of JP2K is particularly significant at low bit rates (say, compression ratios > 10:1).

†School of Electrical Engineering and Telecommunications
The University of New South Wales, Sydney, NSW 2052, Australia.
‡Btexact Technologies, Adastral Park, Martlesham Heath, Ipswich, UK.

The philosophy behind JP2K is that an image can be encoded once (perhaps to the loss-less level) and that it is then up to the decoder (or a transcoder) to extract a sub-set of the bit-stream to reconstruct an image of the required spatial resolution and quality. To enable this functionality JP2K has a highly structured bit-stream, with a main header that describes the image and coding parameters used, and then a series of packet headers that describe the data. The packet header defines the sub-set of wavelet coefficients that are contained in the packet. The terminology used is as follows:

- *Tile*: consists of a whole image or a rectangular (non-overlapping) sub-image;
- *Component*: (normally) a single colour plane of an image;
- *Decomposition level*: a collection of wavelet sub-bands that have the same span with respect to the original samples, i.e., they are from the same resolution;
- *Code-block*: a rectangular grouping of wavelet coefficients from the same sub-band and tile-component;
- *Precinct*: a sub-division of a tile-component within a resolution; and
- *Layer*: a collection of encoded bit-planes, or sub-bit planes, from one, or more, code-blocks of a tile-component. Layers have an order for encoding and decoding that must be preserved.

The data contained in each packet in the bit-stream is from one layer of one decomposition level of one component of an image tile. For further details of the bit-stream syntax and the JP2K coding and decoding process, refer to [2] or the draft international standard [7].

Region of interest (ROI) coding is important in applications where certain parts of an image are of a higher importance than the rest of the image. In these cases the ROI is decoded with higher quality and/or spatial resolution than the background (BG). Example applications include:

- Client/server applications where the server initially transmits a low quality/resolution version of an image. The client then selects an area of the image as a ROI and the server transmits only the data needed to refine (i.e., improve the spatial resolution/quality) of that ROI [4]. This is a useful feature to offer when browsing image databases, particularly when the stored images are high resolution and/or large (say > 2 Mega-pixels),

as it means the client need not download the whole image at the highest resolution;

- Face images. When browsing a digital photograph album it is often the case that we are looking for, or most interest in, the people/faces in those photographs. Using an automated face detection algorithm the region(s) of an image that contain faces can be coded as ROI's and therefore stored with more accuracy than non-face sub-images. Again, this can be used in a client/server type application for image browsing or can be used to reduce the number of bits required to store face images on a digital camera;

The results presented in this paper are primarily concerned with the last application, where the region of interest is fixed and predefined for each image.

II. Region of Interest Coding in JPEG 2000

There are three mechanisms available in JP2K to encode and decode images with varying spatial detail: tiling; code-block selection; and coefficient scaling. These are described in more detail below. The description below is limited to single component, i.e., grey-scale, images. However, it is straightforward to extend the discussion to multi-component (usually colour) images as JP2K essentially encodes each component independently (with an optional colour transform) [2].

A. Tiling

The basic unit to be encoded in JP2K is an image tile. An image can be coded as a single tile or can be partitioned into rectangular, non-overlapping, sub-images and each tile coded independently. Although the main purpose tiling is to reduce the local memory required to perform the discrete wavelet transform (DWT), it can also be used as a mechanism to access spatially distinct areas of the images and to code them with variable quality as required. Each packet in JP2K belongs to a particular tile, therefore when the bit-stream is parsed, the tiles relating to the desired ROI can be decoded to a higher quality, i.e., by decoding more layers for the ROI tiles than the background tiles.

B. Code-block Selection

Code-block selection involves parsing a JP2K bit-stream (encoded to a lossless or visually lossless level) and extracting the packets that contain the code-blocks required to decode the ROI. JP2K packets contain code-blocks belonging to one layer of one precinct at one resolution (decomposition level). In Part I of the JPEG 2000 standard code-blocks must have a fixed size at each resolution, that size ranging from 4x4 to 64x64 (having a maximum area of 4096). Note that as the code-blocks are of a fixed size, they represent an increasing spatial extent at higher levels (i.e., from lower resolution packets). In addition, the decoder may also require code-blocks adjacent to the ROI in order to correctly perform the inverse discrete wavelet transform (IDWT) as the wavelet filters are of finite length and require all of the wavelet coefficients for the whole length of the filter to calculate IDWT. As was the case for tiling, the decoder extracts more layers for the packets related to the ROI than for the background. Alternatively, a transcoder can re-order the bit-stream so that packets related to the ROI appear first in the bit-stream and so are decoded first by a standard JP2K decoder.

C. Coefficient Scaling

In Part 1 of the JP2K standard the method of coefficient scaling used is called the *maxshift* algorithm. This algorithm is described in detail in [4], [5], and [7], but an outline is given here for completeness. The maxshift algorithm proceeds as a five-step process:

1. Generate ROI Mask: Determine the set of wavelet coefficients that belong to the ROI. The mapping of the ROI from the spatial domain to the wavelet domain is dependent on the wavelet filters used and is simplified for rectangular and circular regions;
2. Find scaling value, s: As illustrated in the upper part of Figure 1 the magnitude of the largest wavelet coefficient *not* in the ROI, i.e., contained in the background, is found;
3. Scale down Background coefficients by s: As illustrated in the lower part of Figure 1 all of the wavelet coefficients that are *not* part of the ROI are scaled down by $(s + \Delta)$, where Δ is a small constant. This makes all of the wavelet coefficients in the background have a magnitude < 1. Therefore, the decoder just has to scale up (by $s + \Delta$) all coefficients that have a magnitude < 1, no extra information about the shape of the ROI being required;
4. Write s to the bit-stream;
5. Apply bit-plane entropy coding as usual.

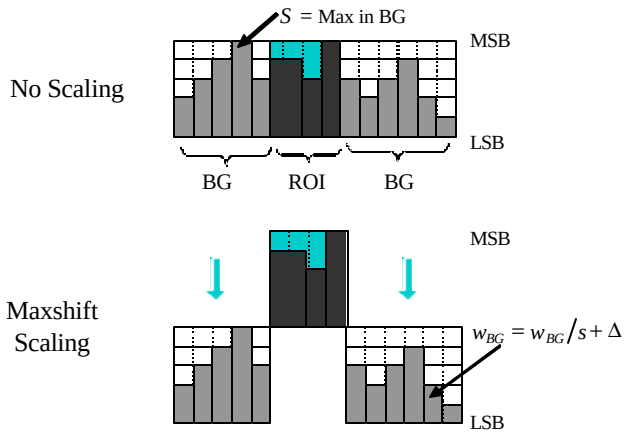

Figure 1. The Maxshift scaling process.

Unfortunately, coefficient scaling reduces coding efficiency, as code-blocks that lie on the boundary of the ROI have to be coded twice: once for the ROI with a zero background, and once for the background with a zero ROI. This overhead typically increases losslessly coded bit-streams by 4-5% [4], [5]. In addition, coefficient scaling also increases the required dynamic range for JP2K. However, when coefficient downscaling is used, as in part I

of JP2K, if there is any clipping, the information lost will be the least significant bits of the background.

III. Experimental Results

In order to compare the three possible methods of ROI coding and to investigate the effect of a number of JP2K parameters on ROI coding performance a set of six images were selected. The images used were from the JP2K test set, they are all 8 bits per pixel (bpp) grey-scale, and represent examples from various types of imagery. There are three *natural* images, i.e., captured with a conventional camera (and therefore diffraction limited): Bike, Café, & Woman; one *aerial radar* image: Aerial2; and two *compound* images, i.e., consisting of text, photographs, and graphics: Cmpnd1 and Chart. Unless otherwise stated progressive, i.e., layered, bit-streams are generated and decoded at a number of bit rates between 0.03125 and 2 bpp. This represents compression ratios between 256:1 and 4:1.

Image distortion is measured as average peak signal-to-noise ratio (PSNR) over all six images, i.e., mean square error (MSE) is averaged over all six images and then PSNR is calculated. When the image to be encoded contains a ROI, PSNR is calculated for the ROI alone and over whole image (for the ROI and the background). For all of the ROI experiments a five level DWT is used and all of the coefficients from the lowest level are included in the ROI. This means that when an image is decoded a low quality rendition of the background is initially decoded; this not only gives the ROI some context, but also simulates a bit-stream that might be required for a client/server application. All ROI coding is compared to JP2K with its default parameter settings, i.e., no ROI, five level DWT, 64x64 code-blocks, 20-layers, layer progressive bit-stream etc. All experiments were completed using version 4.1a of JJ2000, the JP2K (Part 5) reference implementation produced by Canon, EPFL, and Ericsson written in Java.

Figure 2(a) shows the effect of decreasing the size of the tiles used to code the images on rate-distortion performance. It is clear to observe a decrease in compression efficiency with decreasing tile size. This decrease is particularly significant at bit rate < 1bpp where there is as much as a 5dB decrease in PSNR for 128x128 tiles and a 10 dB decrease for 64x64 tiles, at the same bit rate, compared to encoding the image as a single tile. The reasons for the reduction in compression efficiency are that smaller tiles reduce the number of decomposition levels in the DWT and this also forces smaller code-blocks to be used in the sub-bands that are smaller than the desired code-block size. In addition, using tiling at low bit rates (say < 1 bpp) results in clearly identified *block artefacts* in the images at tile boundaries that significantly detract from the visual quality of the decoded images. It should be noted that the visibility of these block artefacts could be significantly reduced using an adaptive filter as a post-processing operation after decompression. However, this adds significant complexity to the decoder.

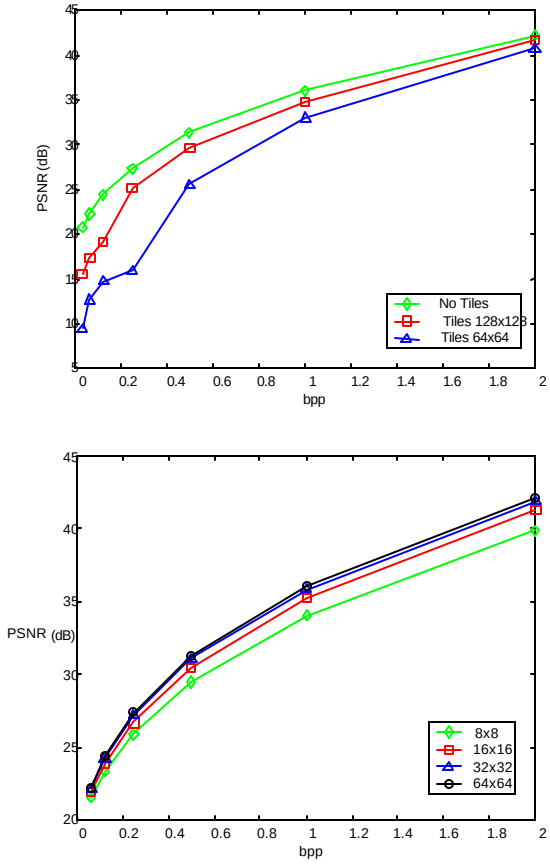

Figure 2. (a) Rate-distortion performance with reducing tile size, (b) with reducing code-block size.

Figure 2(b) shows the effect of decreasing the size of the code-blocks used to code the images on rate-distortion performance. Code-blocks can be reduced from the default 64x64 to 32x32 with only minimal decrease (< 1dB) in PSNR, for a given bit rate. Significant loss in compression efficiency is only observed when the code-blocks are reduced to below 16x16 and when the bit rate is high (say > 0.5 bpp). Therefore, it can be seen that reducing code-block size is a far more effective way of decreasing the spatial size of the coding units in the JP2K bit stream than introducing image tiles. However, image tiles are useful for reducing the local memory buffering requirements to perform JP2K, which is important for inexpensive hardware implementations in applications such as digital cameras.

Figure 3 and Figure 4 illustrate the effect of reducing the size of the ROI on rate-distortion performance. Results presented are for a ROI that is rectangular with a top left hand corner in the centre of the image and size, as a proportion of the total image area, of ¼, 1/8, and 1/16. The maxshift algorithm is used to encode the ROI and in Figure 3(a), (b) and Figure 4(a) the code-block size is 32x32. In Figure 4(b) the code-block size is 16x16. The JP2K bit-streams produced are layer progressive and so increasing rate illustrates the effect of a decoder generating an image of increasing quality as more of the bit stream is received. It can be seen that reducing the size of the ROI decreases the

bite rate at which the ROI is received in full detail, i.e., it increases the speed of ROI refinement. When the ROI is ¼ of the image size the ROI is not received until the rate is above 1bpp, whilst when the ROI is 1/16 of the image size it is has been fully received at 0.25 bpp. This illustrates an approximately linear relationship between ROI size and the rate required to fully decode the ROI. The size of the ROI has a complementary effect on the background refinement, as once the ROI has been received in full, code-blocks related to the background will then be present in the bit-stream. Therefore, as shown in Figure 4(a), when the ROI is 1/16 of the image size, the rate-distortion curve for the whole image is close to that for the image coded with no ROI, especially at rates > 0.5bpp. At low bit rates (say, < 0.125 bpp) with the larger ROI (>1/8 image size) it is possible that the PSNR for both the ROI and the whole image will be below that for the image coded with no ROI. In these cases in can be concluded that the overhead associated with ROI coding is too great and it is therefore more efficient to transmit the image encoded with no ROI information.

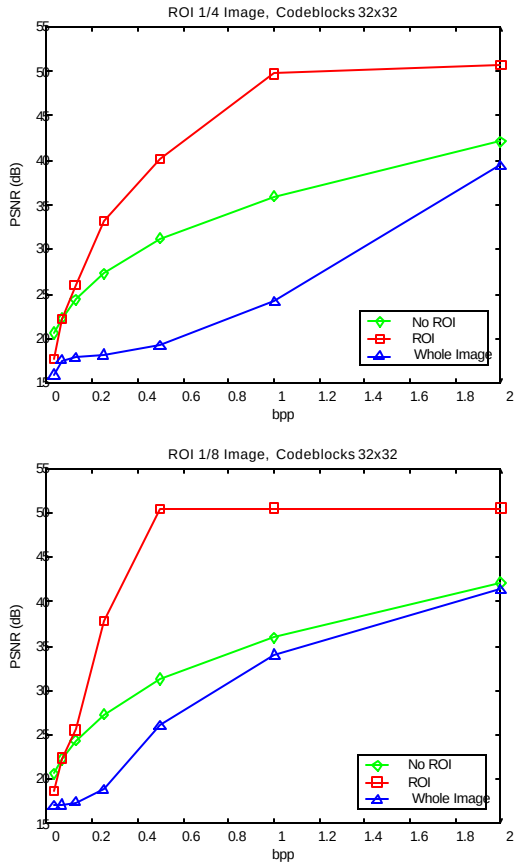

Figure 3. (a) Rectangular ROI ¼ of image size. (b) Rectangular ROI 1/8 of image size.

Figure 4(b) illustrates the inter-dependence between ROI size and code-block size. This inter-dependence occurs because of the trade-off between the ROI overhead and coding efficiency as code-block size is reduced. In other words, as the code-block size is decreased the ROI overhead also decreases (as, on average, smaller code-blocks will contain more of the ROI and less zero coefficients from the background), but the coding efficiency will decrease (as was shown in Figure 2(b)). This leads to the observation in Figure 4(b) that decreasing the code-block size (in this case to 16x16) decreases the bit rate required to refine the ROI, but the rate-distortion performance on the rest of the image is reduced. Therefore, in applications where the ROI is of primary importance smaller code-blocks (say 16x16) should be used. This is especially the case at reduced bit rates (say < 0.5 bpp). However, if the quality of the background is also of some importance, or the desired bit rate is higher (say ≥ 1 bpp) then larger code-blocks should be used (say 32x32).

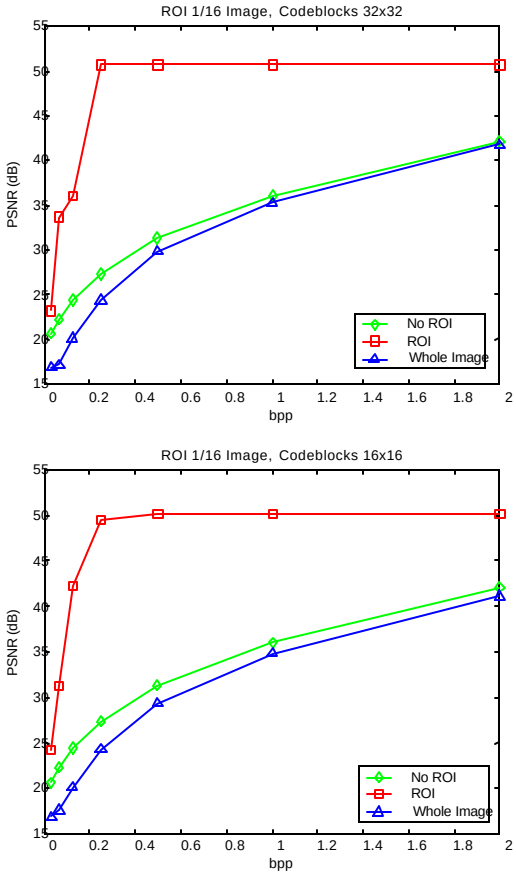

Figure 4. (a) Rectangular ROI 1/16 of image size. (b) ROI 1/16 of image size, 16x16 code-blocks.

Figure 5(a) illustrates the rate-distortion performance of code-block selection decoding an image with a ROI 1/16 of the overall image size. Comparing Figure 5(a) to Figure 4(a) shows that code-block selection has inferior rate-distortion performance (i.e., slower refinement) on both the ROI and the background than the maxshift algorithm. However, code-block selection does not require the ROI to be encoded into the bit-stream when the image is originally encoded. This allows the user the added flexibility to specify the ROI at decode time, which is vital to client/server applications.

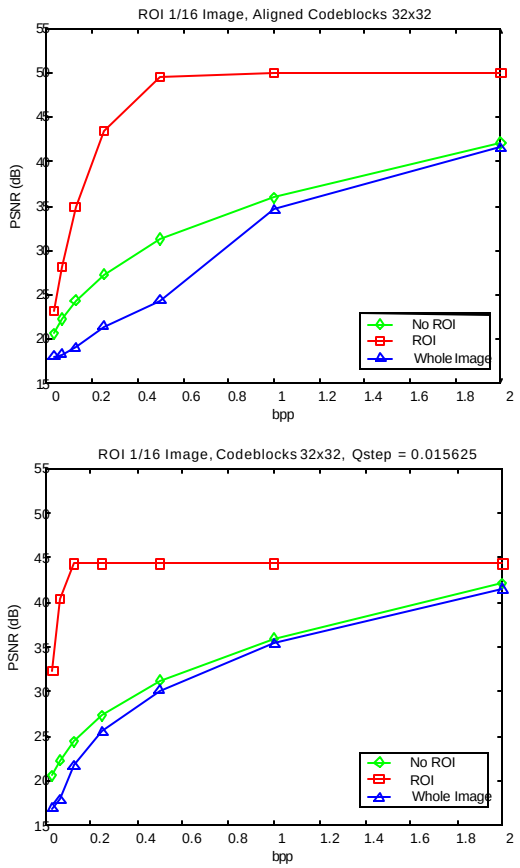

Figure 5. (a) ROI via code-block selection (b) Effect of doubling quantiser step size.

A potential problem common to both the maxshift and code-block selection methods is that the ROI must be fully decoded before the background can be significantly improved. Observation of the progressively decoded bit-stream shows that the high quality layers (associated with the least significant bits of the wavelet coefficients in the ROI) add very little to the visual quality of the ROI. Therefore, it would seem to make sense to terminate refinement of the ROI before we reach the last layer(s) and instead send lower layers (more significant bits) from the background. This is easily achieved in the code-block selection method by appropriately decoding/ordering packets in the bit-stream. However, the maxshift algorithm fully encodes the ROI layers before encoding any layers from the background. There are two possible solutions to this problem:

1. The general coefficient scaling algorithm, described in [5], can be utilised. However this is currently not part of the JP2K standard and because coefficients from the background can not be guaranteed to be smaller than coefficients from the ROI a mask must be transmitted to the decoder to indicate which coefficients need re-scaling. Mask transmission implies a further overhead on the bit-stream and therefore a reduction in compression efficiency. This overhead may be significant for an ROI of arbitrary shape, thus negating the benefits of the general scaling method;
2. The step-size of the (uniform) quantiser used to encode the wavelet coefficients can be increased. This has the effect of not coding the least significant bits (layers) of the wavelet coefficients. However, this reduction in accuracy is applied over the whole image and cannot be reduced later in the bit stream. Figure 5(b) shows the effect on rate-distortion performance of doubling the default quantiser step-size. Clearly the ROI is decoded to a lower quality, however in this case it is still coded to the visually lossless level. The ROI is now fully transmitted when the rate is 0.125 bpp (compared to 0.25 bpp for the default quantiser of Figure 4(a)) and results in a corresponding increase in the quality of the background at bit rates > 0.125 bpp. Therefore, it would appear that controlling quantiser step size in the maxshift method is the most appropriate technique for lossy ROI coding.

As already mentioned, Part 1of the JP2K standard is limited to having the same size code-blocks at each level of the DWT (resolution). This means that code-blocks in higher level sub-bands of the DWT relate to an increasing spatial size, e.g., at level 1, a 16x16 code-block relates to a 32x32 image block; at level 2, a 16x16 code-block relates to a 64x64 image block etc. The effect of the increasing *spatial* size of code-blocks results in a gradual reduction of image quality in the regions around the ROI. This also explains why the ROI refinement with code-block selection is slower than with the maxshift algorithm. Initially code-blocks (from high levels in the DWT) are being transmitted that relate not only to the ROI, but also to regions adjacent to it. Therefore, code-block selection may not suitable for low bit rate applications (say < 0.5bpp). The superiority of the maxshift algorithm over code-block selection will dependent upon the requirements of the application, i.e., whether the ROI should be embedded in the bit-stream, the shape and size of the ROI, the target bit rate, and the viewing preferences of the users of the system. For client/server applications it is essential to be able to extract any ROI from an encoded image, in which case code-block selection is the best method to use. This would be the case in an automatic ROI extraction system [8] where the size and shape of the regions of interest can be derived from a wide range of sources. However, in the application examined in this paper: facial images; the ROI is known and fixed. Therefore, it is desirable to have the ROI embedded in the bit-stream using the maxshift algorithm. In addition, the ROI is of primary importance and we desire to receive the ROI as early as possible in the bit-stream (which is equivalent to a low bit rate application) and so use 16x16 code-blocks and a quantiser step size of 0.03125 (four times the default).

JP2K JP2K ROI

Figure 6. Comparison of default JP2K and maxshift ROI coding on a face image (0.125 bpp).

IV. FACIAL IMAGES

Figure 6 demonstrates the improved quality of the face portion of the image over the default (no ROI) JP2K whilst Figure 7 illustrates a marginal improvement in the quality of the face portions of the image when there are two faces present in the image. With more than one ROI the overhead associated with the maxshift algorithm can negate the increase in quality observed on the faces. In both of these examples, the face part of the image was selected by hand and encoded in JP2K as a circular ROI.

JP2K JP2K ROI

Figure 7. Comparison of default JP2K and maxshift ROI coding with two face ROI's (0.25 bpp).

V. CONCLUSIONS

JPEG 2000 offers significant improvements over previous image compression standards not only in terms of compression performance, but also coding flexibility. However, to fully utilise the features available in JPEG 2000 requires an understanding of both the encoding algorithm and the parameter set used to control it. This is particularly important in applications where the main goal is not maximum compression, but other factors such as memory usage, region of interest (ROI) coding, or browses facilities. This paper has outlined some of the trade-offs in relation to ROI coding and has highlighted the following points:

- The efficiency of ROI coding is dependent upon the bit rate, ROI size, and relative ROI to background visual importance. In particular, 32x32 code-blocks should be used when the ROI is large, the bit rate high, or when the background is visually important. However, 16x16 code-blocks should be used when the ROI is smaller, the bit rate is lower, or when the ROI is of primary visual importance;
- Tiling is not an efficient method of ROI coding, unless memory constraints are of primary importance. ROI coding using coefficient scaling is suitable for applications requiring an ROI of arbitrary shape that is embedded in bit-stream, such as face images. It should only be used when the ROI is less than ¼ of the total image area and to prevent ROI over-coding the quantiser step-size should be increased. ROI coding using code-block selection is suitable for applications requiring the extraction of an arbitrary ROI from the coded bit-stream, such as client/server applications.

VI. REFERENCES

[1] W. B. Pennebaker and J. L. Mitcell, "JPEG: Still Image Compression Standard," Van Nostrand Reinhold, 1993.

[2] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG 2000 Still Image Coding System: An Overview," *IEEE Transactions on Consumer Electronics*, Vol. 46, No. 4, pp. 1103-1127, November 2000.

[3] D. Santa-Cruz, T. Ebrahimi, J. Askelof, M. Larsson, and C. A. Christopoulos, "JPEG 2000 Still Image Coding Versus Other Standards," Proceedings of SPIE, Vol. 4115.

[4] D. Santa-Cruz, T. Ebrahimi, M. Larsson , J. Askelof, and C. A. Christopoulos, "Region of Interest Coding in JPEG 2000 for Interactive Client/Server Applications," Proceedings IEEE 3rd Workshop on Multi-media Signal Processing, pp.389-394, September 1999.

[5] C. A. Christopoulos, J. Askelof, and M. Larsson, "Efficient Methods for Encoding Regions of Interest in the Upcoming JPEG 2000 Still Image Coding Standard," *IEEE Signal Processing Letters*, Vol. 7, No. 9, pp. 247-249, September 2000.

[6] D. Taubman, "High performance Scalable Image Compression With EBCOT," *IEEE Transactions on Image Processing*, Vol 9, No. 7, pp. 1158-1170, July 2000.

[7] M. Boliek, C. Christopoulos, and E. Majani (Eds), "JPEG 2000 Part 1 Final Draft International Standard," (FDIS15444-1), ISO/IEC JTC1/SC29/WG1, N1855, August 2000.

[8] F. Stentiford, "An Estimator for Visual Attention through Competitive Novelty with Application to Image Compression," Picture Coding Symposium, pp. 25-27, April 2001.