Multi-component Document Image Coding Using Regions-of-Interest

Xiao Wei Yin, Andy C. Downton, Martin Fleury, and J. He

Department of Electronic Systems Engineering, University of Essex, UK
{xwyin,acd,fleum,jhe}@essex.ac.uk

Abstract. Region-of-Interest (ROI) techniques are often utilized in natural still-image coding standards such as JPEG2000 [1]. In contrast, document image coding typically adopts multi-layer methods [2], using a carefully selected algorithm for each layer to optimize overall performance. In this paper, an ROI-based method is proposed for multi-component document image coding, where rectangular textual ROI's are easily extracted using standard document image analysis techniques. Compared to multi-layer methods, the method is simpler and scalable, while preserving comparable visual quality at equivalent PSNR.

1 Introduction

Historically, the most common format for document images has been binary for reasons of efficient storage, leading to the development of binary document image coding standards such as JBIG1 and JBIG2 [3], a token-based compressor. However, as demand for higher image quality has grown and the range of digitized documents increased, gray-scale and color document image representations have become common, although these increase storage space and/or transmission time. Hence, it is now essential to design document image coding algorithms that can compactly represent multi-component document images.

The primary content of multi-component document images typically consists of text, lines and drawings, and high resolution is always required to display this foreground information. The residual can be regarded as background, and for transmission purposes is a less important region, which can be displayed later and at lower resolution than the foreground, as is also considered acceptable in [4]. The majority of color document image compression algorithms use segmentation-based multi-layer methods to meet these requirements. One example, DjVu [2], defines three layers: mask layer, foreground layer, and background layer. The mask layer specifies the shape of text and high-contrast lines, distinguishing which pixels should be coded using the foreground or background coding algorithms. The DjVu foreground layer defines the color detail within the mask layer, while background texture outside the mask is coded using wavelet-coding techniques. In [2] Figure 3, examples of text compressed by the DjVu technique are shown for an 'XVIIIth Century book' (historical font) and the 'US First Amendment' (handwritten), along with contemporary newspaper, magazine and scientific articles.

S. Marinai and A. Dengel (Eds.): DAS 2004, LNCS 3163, pp. 158–169, 2004.

The independent coding of foreground mask and background layers allows high subjective document image quality to be maintained by prioritizing the bit-budget towards the foreground layer. However, different coding methods are applied to each of the three layers, effectively coding each image three times. For example, DjVu [2] uses the JB2 algorithm, a variant of JBIG2 and hence a token-based compressor, for the mask layer at 300 dpi, and the IW44 wavelet encoder for the background layer at 100 dpi. This increases the total complexity and coding delay, and constrains potential application areas. It also introduces redundant data, leading to sub-optimal objective performance in both lossless and 'lossy' coding modes. Our experimental work [5] also shows that the mask layer typically occupies at least half of the total file size, limiting scalability.

In this paper, an alternative approach is proposed that uses wavelet coding for both foreground and background regions, but which bit-plane shifts to produce a high-quality foreground image. The approach has been tested on archive and historical documents. The method uses a simple rectangular text block detection process to segment the ROI, which is then coded with a very small bit overhead, compared to the DjVu mask layer. This approach is similar to the JPEG2000 ROI method, except that the concept of the system automatically determining the ROI map is lacking in JPEG2000, since in natural images there is no general criterion that can be specified to identify ROI's. In contrast, rectangular regions predominate in 'city block' document formats, although, in general, the ROI method is not restricted to rectangular regions as, just as in JPEG2000, an arbitrary region can be defined by its coordinates. Compared to multi-layer methods, the proposed coding method is simple, can include both 'lossy' and near lossless representations in a single stream, and supports both Signal-to-Noise (SNR) and resolution scalability with comparable visual quality at equivalent scale. Where text regions are small and irregular in some way, then a token-based compressor is limited by the reduction in repeated patterns. However, an example document which is all text is included in the tests, compensating for this weakness. The major advantage of the ROI approach is in encoding documents with regions of differing contrast and varying text types, whereas multi-layer methods are more suited to documents with high-quality text and embedded high-quality continuous tone illustrations, as occur in some magazines.

2 Text Region Detection

Archive documents are a typical example where a richer representation of a color document image is required to convey not only the textual content of the document (which could be satisfactorily represented in binary or even as encoded characters) but also its feel and context. Fig. 1 shows an example document image from an index card archive of Lepidoptera (butterflies and moths) at the UK Natural History Museum, with superimposed rectangular textual ROI's. The full Lepidoptera archive consists of several hundred thousand cards, now searchable over the Internet [6].

The format of archive index cards consists of several independent blocks of text, and each block consists of one or more logically related text fields, as shown in Fig. 1. Blocks retain a fairly consistent mutual layout over a complete archive, but the layout of text fields within each block is not strictly fixed. Nor are there any tabular guidelines

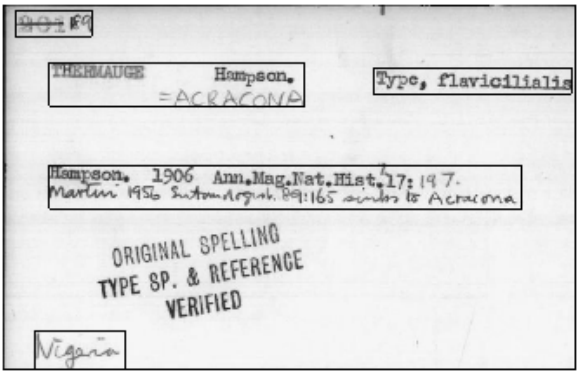


Fig. 1. Example Lepidotera archive index card, showing foreground (detail) and background regions.

defining fixed block boundaries. The X-Y cuts algorithm is, therefore, an appropriate segmentation algorithm for this class of document image structure.

Pixel smearing [7], with a threshold sufficient to join adjacent text characters but not adjacent horizontal words or vertical lines, is applied as a pre-processing non-linear low-pass filter to each archive card image. The X-Y cuts algorithm [8] is then applied to the set of archive cards. The algorithm extracts and stores the contents of each index card into a hierarchical tree structure (the so-called X-Y tree), consisting of text blocks, lines and words. The level of the cut is dependent on the white space: conventionally, the space between blocks is greater than between lines and words. The first level cut thus separates horizontal blocks (which may contain several lines of text) based upon their vertical spacing, the second level segments lines vertically, and the last level cut separates words horizontally within each line, using binary connected component bounding boxes. The extracted contents stored in the X-Y tree thus follow a sequence where the top level of the tree stores blocks while the bottom level stores individual words. Alternatives techniques are widely reported in the document analysis literature for segmenting document images with different characteristics from the Fig. 1 example, such as the other examples considered in our experiments (refer forward to Figs. 4 and 5). In Fig. 1, the `ORIGINAL SPELLING` etc. stamp is a redundant artifact of no archival value. Digital removal of such features before transmission is described in [9], and, hence, is not considered herein.

3 Generating the Region of Interest Map

Because each text ROI is rectangular in shape, the calculation of the ROI map can be dramatically reduced, adapting an existing algorithm [10] for that purpose. As in a conventional wavelet-decomposition, once the ROI map is generated at the image scale, the identification process is recursively repeated at each lower subband, until the predefined maximum level depth is reached. The exact mechanism for generating the ROI map is related to the wavelet algorithm chosen. The well-known 9/7-tap filter serves as an example to explain how to generate the interest map.

Suppose there is a one dimensional (1-D) ROI pixel sequence, denoted as $X(n)$. After the 9/7 tap wavelet transform, the low- and high-frequency wavelet coefficient sets, denoted respectively $L(.)$ and $H(.)$, that are responsible for the reconstruction of $X(n)$ are:
if n is even, with $m = n/2$

$$L(.) = \{L(m-1), L(m), L(m+1)\}$$
$$H(.) = \{H(m-2), H(m-1), H(m), H(m+1)\}$$

else if n is odd, let $m = (n\text{-}1)/2$

$$L(.) = \{L(m-1), L(m), L(m+1), L(m+2)\}$$
$$H(.) = \{H(m-2), H(m-1), H(m), H(m+1), H(m+2)\}$$

Because the 2-D wavelet transform is separable, a 2-D mapping can be identified from 1-D mappings in the horizontal and vertical direction, which correspond to $L(.)$ and $H(.)$, depending on subband level. For any one pixel in the ROI, let $x_{L(.)}$ and $y_{L(.)}$ represent the sets of Cartesian coordinates from the coefficient set $L(.)$ (and similarly for $x_{H(.)}$and $y_{H(.)}$), then four displaced rectangular regions can be identified as their direct product coefficient coordinate sets:

$$x_{L(.)} \otimes y_{L(.)}, x_{L(.)} \otimes y_{H(.)}, x_{H(.)} \otimes y_{L(.)}, \text{ and } x_{H(.)} \otimes y_{H(.)}. \quad (1)$$

Each pixel in an ROI generates four similar wavelet coordinate coefficient sets, and the four contributing subband regions are the union of the corresponding coordinate coefficient sets of all pixels.

Rather than applying (1) to each pixel in the ROI, by determining the wavelet coefficients corresponding to just the top left and bottom right corner pixels of each rectangular ROI [10], the complete set of displaced rectangular regions can be generated in a simplified manner. Let$(x_m, y_n), (x_k, y_l)$denote respectively the top left and bottom right corners of the text region. Apply (1) to each of these two corners, with ($m_{\min}$, $n_{\min}$) representing in turn the coordinate of the top-most left coefficient in each of the four subband regions generated by (x_m, y_n), and $(k_{\max}, l_{\max})$ similarly corresponding in turn to the bottom-most right coefficient of each of the four subband regions generated by (x_k, y_l), then, for each of the four subband regions, the coordinates correspond to the set

$$\{(x_h, y_p) | m_{\min} \leq h < k_{\max}; n_{\min} \leq p < l_{\max}\} \quad (2)$$

Figure 2 is an example showing a 3×3 rectangular coefficient region, with (1) performed on the two corner coefficients to generate two groups of four sets in each subband region. The extremes of these sets identify the desired subband regions.

4 Region of Interest Technique

Once the ROI map is generated, all the wavelet coefficients within that region need to be included in the compressed bitstream. Several bit-plane shifting algorithms are available [10] for making these coefficients appear as early as possible. Alternatively

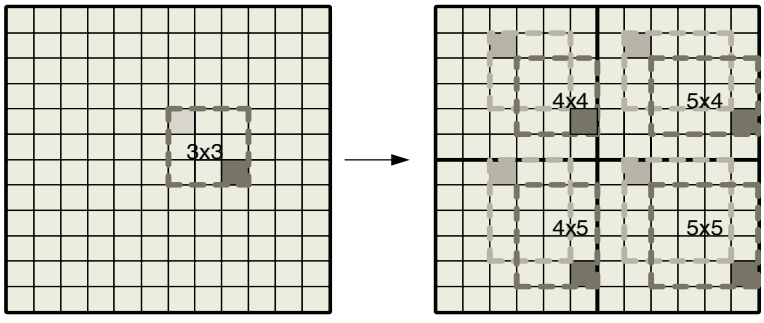


Fig. 2. Using two corner pixels to generate subband regions, showing a single-level subband decomposition.

since the ROI algorithm performance is encoder dependent, Park & Park [11] avoid bit-plane shifting for the SPIHT encoder [12], as by mixing ROI and non-ROI coefficients in the same bit-plane through shifting, some correlations, which are otherwise present, cannot be used to reduce the bit-rate. The research in [11] utilizes: 1) a parent of ROI (PROI) mask; and 2) the use of per-bit-plane multi-lists, which retain information on tested coefficients that lie outside the ROI, thus preventing information wastage in later encoding rounds.

The work reported here uses a similar method to [11], but with some further enhancements, and adaptations for document coding, where neither an ROI, nor PROI map is needed, because each ROI is a rectangular box, and can be represented simply as two corner coordinates (Section 3). Instead, a dynamically applied function determines (using the stored corner coordinates) whether the current coefficient or set of coefficients is or contains an ROI coefficient. This modification considerably reduces the memory that would otherwise be needed if ROI and PROI maps were to be used. To avoid excessive modification of the SPIHT algorithm, a single list was used rather than the multi-lists of [11]. This is achieved by simply adding a coefficient bit-plane level indicator, in addition to the coefficient co-ordinates normally held in SPIHT's significance lists. The indicator records the bit-plane level at which a coefficient could become significant during later processing rounds. Neither of these two changes affects the performance of the SPIHT algorithm.

5 ROI Algorithm Description

Given a wavelet-transformed image, X, let $n_{\max} = \lfloor \log_2(\max(\{c_{i,j}\})) \rfloor$ $n_{\max} = \lfloor \log_2(\max(\{c_{i,j}\})) \rfloor$, $c_{i,j} \in X$. In the same manner as SPIHT, define sets $D(b)$as all descendants for coefficient b, and $L(b)$ as all descendants except the direct descendants. Also, define three ordered auxiliary lists: (i) LIP to contain insignificant pixels; (ii) LIS to contain insignificant sets; and (iii) LSP to contain significant pixels/coefficients. Entries in LIS can be of type A or B (corresponding to D or L type descendants).

Two parameters, which avoid shifting, are user definable; these are p and r, as shown, together with matching bit-planes thresholds n_{max} to $n_{0,}$in Fig. 3. The first

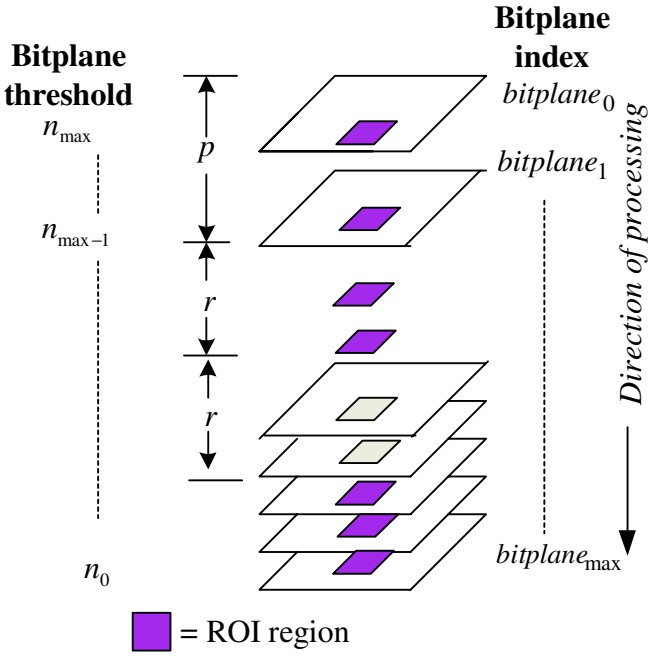


Fig. 3. This figure shows that after pth bitplane encoding, the ROI will be encoded r bitplanes earlier than the rest of image.

p bitplanes, indexed 0 to $p-1$, are encoded using SPIHT over the whole image. Subsequently, the ROI only is encoded for r bit-planes, indexed p to $p+r-1$, and the resulting bits are transmitted. Then, the complete transform image, apart from the ROI, is encoded for the same r bit-planes, reusing significance information from the ROI-only encoding rounds. Finally, the remaining bit planes, indexed $p+r$ to *max*, are encoded, using SPIHT over the whole image. For bitplane q, when $p \leq q < p+r$, a modified SPIHT is applied, in a similar manner to Fig. 7 in [11]. However, unlike [11], all decisions about membership of an ROI or PROI are replaced by a 'judgement' function:

```
Judge_RoI((i,j),type)
{
case type:

  coefficient:check whether c_{i,j}belongs to RoI;
              if so, return true;
  A: check whether any member in D(c_{i,j}) belongs to RoI;
     if so, return true;
  B: check whether any member in L(c_{i,j}) belongs to RoI;
     if so, return true;

}
```

The judgement is made either on a single coefficient in a LIP or LSP, or on coefficient sets of type A or B in a LIS. Initially (in round 0), all coefficient bit-plane indicators are set to zero. In each bitplane round, once a coefficient's significance has been tested, if it is significant then its indicator value will be increased. For example in bitplane q, suppose $D(i,j)$ is found to be significant and $Judge\_RoI((i,j),A)$ is true. Then, for each of $c_{2i,2j}, c_{2i+1,2j}, c_{2i,2j+1}, c_{2i+1,2j+1}$, and $L(i,j)$, if it belongs to the ROI map, test its significance, and if significant, increment its significance indicator to $q+1$. If a coefficient is not in the ROI map there is no need to test significance, and the significance indicator remains as q. Thus, the algorithm can precisely record in which bitplane round each coefficient becomes significant. In fact, significance counting is not confined to the ROI bitplane rounds, but is used for all bitplane rounds.

For the remaining full bitplanes after the ROI rounds, special treatment should be accorded to the coefficients or sets of coefficients that indicate possible significance. In bitplane q, only those coefficients having a significance indicator equal to q are tested, because it is already known that other coefficients are not significant in this cycle.

6 Experimental Results

DjVu [2] was compared with the proposed ROI method. ROI's were implemented using IW44 , SPIHT [12], and JPEG2000 [11] algorithms. JPEG2000 testing was based on the source code of JJ2000[1], which is recommended on the JPEG official webpage. IW44 is the wavelet encoder used for DjVu background images. Because the parameter r, used to control the relative importance between foreground and background as explained in Section 4, can range from 0 to r_{max}, it was impractical to test the impact of all possible values. Thus, a middle value of 4, and a high value of 8 were respectively chosen. For SPIHT, the equivalent of the max-shift method [10] was also implemented by setting $r = r_{max}$. The main advantage of the max-shift method is that the interest map doesn't need to be sent to the decoder, but, in this case, the information about ROIs is just a few coordinates. Therefore, no obvious advantages for the max-shift method are shown by the test results. The Peak Signal-to-Noise (PSNR) figures are compared in two ways, first for the whole image, and secondly for the ROI areas only.

The test images are chosen from three possible application areas, by considering 1) both Western and Asian images, 2) text only and text-drawing images, and 3) printed text and handwritten text.

The first image used as an example of our results was randomly chosen from the card archives held at the Natural History Museum in London. As shown in Fig. 1, five separate ROI areas were identified. In Fig. 1 and subsequently, $r = r_{max}$, to illustrate the effect more clearly. Two representative rate points are illustrated in the results, a low rate (at 0.075 bpp) and a relatively high rate (at 0.28 bpp). Test results are given in Table 1. From those data, it can be seen that the SPIHT ROI generally gives better objective performance than DjVu. ROI image quality is improved when r is increased, trading off against the background image quality. This can be seen by the comparison between SPIHT_roi(4) and SPIHT_roi(8). The subjective image quality of the SPIHT

[1] Java implementation of JPEG 2000, available at `http://jpeg2000.epfl.ch/` (last accessed xi 14 03)

ROI is comparable with DjVu, as shown by Fig. 4. Although image b in Fig. 4 contains some obvious artifacts, it is also more readable than image a, especially if blown up so that detail is visible. Image c seems more readable than image d, but this is caused by the loss of many details in the background. For the detailed text region (zoomed 300% in the left top corner of the test image), the proposed method is subjectively closer to the original image. Also recall from Section 2 that the stamp would normally be removed before transmission.

Table 1. PSNR comparisons of each ROI method for the archive card example, Fig. 4, the 'Jefferson' letter, Fig. 5, and the Chinese ancient art, Fig. 6.

Image	Bit-rate	Method:	DjVu	IW44	IW44_ roi(4)	IW44_ roi(8)	SPIHT	SPIHT_ roi(4)	SPIHT_ roi(8)	SPIHT_ roi(max)	JPEG2000_ roi(max)
Figure 4	0.075 bpp	Whole	21.97	26.98	22.58	20.85	27.11	27.10	22.47	22.47	25.57 (dB)
		ROI only	18.07	22.35	23.40	23.40	22.30	22.31	24.56	24.56	23.16 (dB)
	0.28 bpp	Whole	31.08	34.91	26.69	21.38	35.19	35.33	30.16	23.09	28.34 (dB)
		ROI only	26.97	30.52	33.45	34.31	30.79	31.00	34.40	34.99	33.08 (dB)
Figure 5	0.039 bpp	Whole	30.20	35.30	35.30	34.29	34.65	34.65	30.53	30.53	13.12 (dB)
		ROI only	26.15	32.01	32.01	31.48	31.25	31.25	32.20	32.20	33.25 (dB)
	0.139 bpp	Whole	40.80	41.03	41.03	37.26	40.76	40.76	40.75	31.52	13.14 (dB)
		ROI only	38.82	39.24	39.24	36.16	39.21	39.21	39.34	40.57	41.15 (dB)
Figure 6	0.028 bpp	Whole	18.66	22.75	21.63	20.15	23.02	23.02	23.00	23.00	23.96(dB)
		ROI only	15.01	19.15	19.17	19.33	19.39	19.39	19.40	19.40	19.38 (dB)
	0.104 bpp	Whole	27.03	31.41	31.41	30.87	33.09	33.09	32.27	32.27	32.82 (dB)
		ROI only	23.29	28.32	28.32	29.11	30.23	30.23	30.44	30.44	30.39 (dB)

Another test image was chosen from the Thomas Jefferson papers at the Library of Congress; test data in Table 1 shows similar results, and the subjective performance is compared in Fig. 5, confirming the results from the archive card image.

An example of Chinese ancient art, which contains a drawing, was also chosen as a test image, and a set of visual comparisons are shown in Fig. 6. Zoomed-in detail shows the advantage gained from using an ROI method for the drawing. In the low-resolution results, the text decoded by DjVu looks sharper than when using the ROI compression method. However, the cost of using DjVu is that the quality of the drawing part degrades more than by the ROI method. This weakness is the result of DjVu's multi-layer method. During segmentation processing, DjVu performs well within the text region, but sometimes relegates some parts of the drawing to the background layer. The background layer is treated as a lower quality layer in DjVu, leading to the quality of the decoded drawing being reduced. In contrast, the ROI method avoids this problem. By selecting individual regions, all the useful regions are well protected, no matter whether they are text or drawing regions. In high-resolution compression, both text and drawing regions in the ROI method look slightly better than DjVu, in spite of some artifacts around the edges. The objective results, Table 1, correspond to the above subjective comments in respect to the high-quality results. The low-resolution results actually show the advantage of the SPIHT over DjVu and not any particular advantage at low resolution for the ROI method. This may be because SPIHT has suppressed some high-frequency artifacts

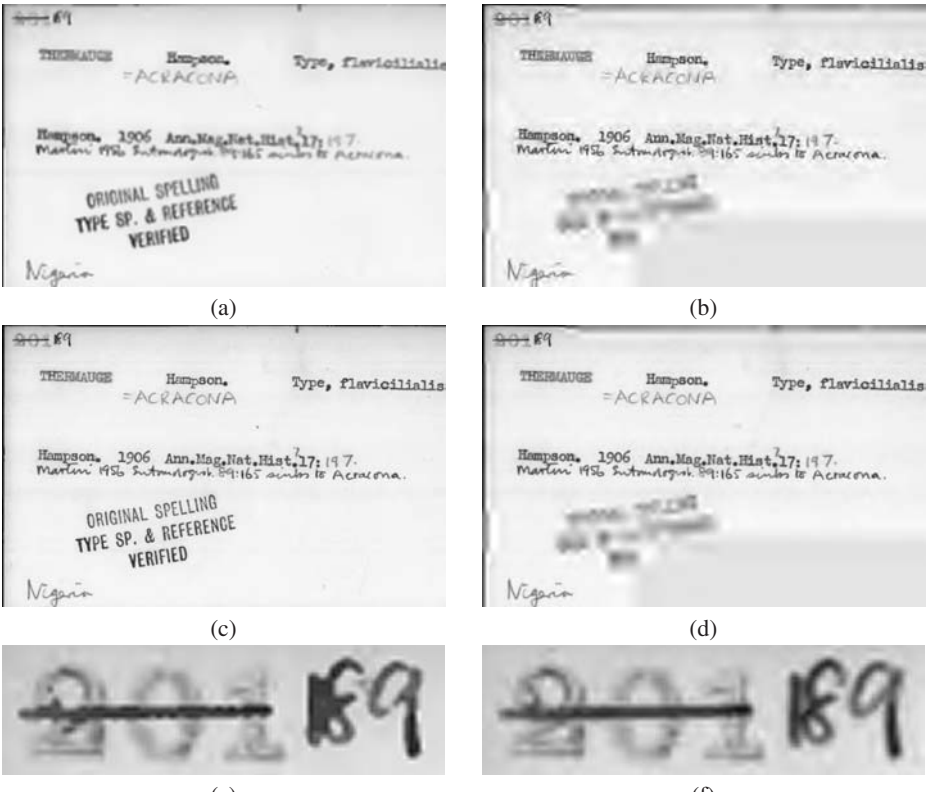
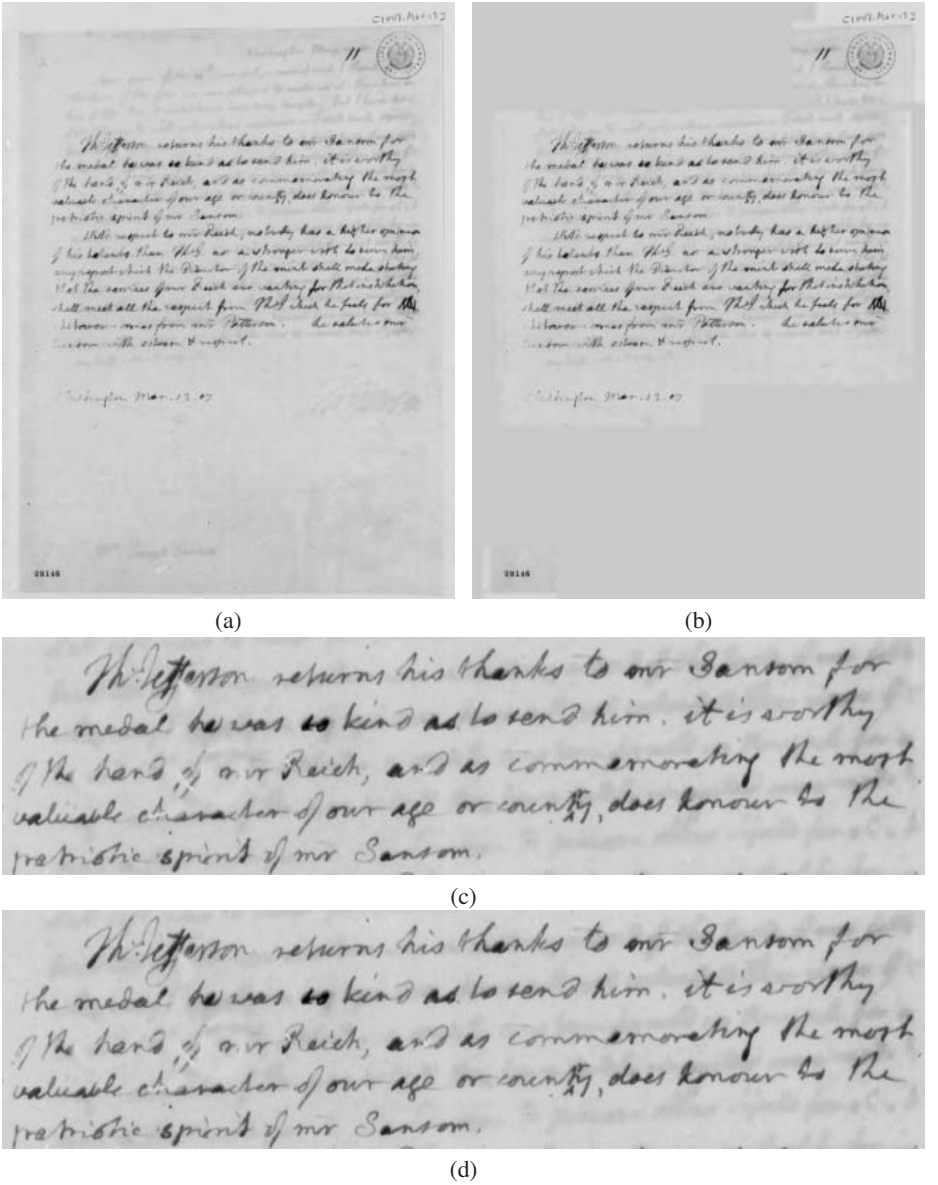


Fig. 4. Visual image quality comparison:(a) DjVu at 0.075 bpp, (c) DjVu at 0.28 bpp (b) SPIHT_roi_max at 0.075 bpp, (d) SPIHT_roi_max at 0.28 bpp, (e) 300% zoom of the top left corner of (c), (f) 300% zoom of the top left corner of (d).

in the background of the image, whereas DjVu has included these high-frequency artifacts in the foreground layer. However, for low-resolution images, SPIHT is as good at the suppression of high-frequency components as the ROI method. The high-frequency artifacts are mostly located at the bottom of the image, where a close inspection shows random fluctuations of image intensity.

7 Conclusions

In this paper, a simple block-based document segmentation algorithm, combined with ROI methods applied to wavelet coding, is proposed as an alternative to current multi-layer document coding methods that apply different coding techniques to each layer. The proposed approach can be applied to a number of wavelet coding algorithms: the paper compares objective PSNR and subjective visual performance of IW44, SPIHT and JPEG2000 implementations with a well-known commercial multi-layer coding algorithm (DjVu). By taking advantage of the simple representation of ROIs using corner coefficients, further optimisation of SPIHT is possible, resulting in a computationally



(a) (b)

(c)

(d)

Fig. 5. Thomas Jefferson Papers Series 1: letter extract (a) generated by DjVu (b) generated by the ROI method (c) zoomed version of (a) (d) zoomed version of (b).

efficient scalable coder with favourable performance at very low bit rates. This SPIHT implementation may be particularly appropriate for use in application areas such as mobile devices, where display resolution, communications bandwidth and/or processing capacity are limited, and direct hardware implementation of the coding algorithm may be required.

Fig. 6. Chinese ancient art: (a) original image (b) DjVu at 0.028 bpp (c) DjVu at 0.104 bpp (d) SPIHT_roi_max at 0.028 bpp (e) SPIHT_roi_max at 0.104 bpp (f) Zoomed in version of (c) (g) Zoomed in version of (e).

References

1. Christopoulos, C., Skodras, A., Ebrahimi, T.: The JPEG 2000 still image coding system: An overview. IEEE Transactions On Consumer Electronics **46** (2000) 1103–1127
2. Bottou, L., Haffner, P., Howard, P.G., Simard, P., Bengio, Y., Le Cun, Y.: High quality document image compression with DjVu. Journal of Electronic Imageing **7** (1998) 410–425
3. Howard, P.G.: Text image compression using soft pattern matching. The Computer Journal **40** (1997) 146–156
4. Haffner, P., Bottou, P., Howard, P.G., Simard, P., Bengio, Y., Le Cun, Y.: Browsing through high quality document images with DjVu. In: Advances in Digital Libaries. (1998) 309–318
5. Yin, X.W., Fleury, M., Downton, A.C.: Archive image communication with improved compression. In: ICDAR 2003. Volume 1. (2003) 92–96
6. Beccaloni, G., Scoble, M., Robinson, G., Pitkin, B.: The global lepidoptera names index, at `http://www.nhm.ac.uk/entomology/lepindex/` (2002)
7. Wong, K.Y., Casey, R.G., Wahl, F.M.: Document analysis system. IBM Journal of Research and Development **26** (1982) 647–656
8. Ha, J., M.Haralick, R., Phillips, I.T.: Recursive X-Y cut using bounding boxes of connected components. In: ICDAR'95. Volume II. (1995) 952–955
9. He, J., Downton, A.C.: Configurable text stamp identification tool with application of fuzzy logic. In: IAPR Workshop on Document Analysis Systems. (2004) Elsewhere in this volume.
10. Christopoulos, C., Askelof, J., Larsson, M.: Efficient methods for encoding region of interest in the upcoming JPEG2000 still image coding standard. IEEE Signal Processing Letters **7** (2000) 247–249
11. Park, K., Park, H.W.: Region-of-interest coding based on set partitioning in hierarchical trees. IEEE Transactions On Circuits and Systems for Video Technology **12** (2002) 106–113
12. Said, A., Pearlman, W.A.: A new, fast, and efficient image codec based on set partitioning in hierarchical trees. IEEE Trans. on Circuits and Systems for Video Tech. **6** (1996) 243–250