Proc. IWSSIP, pages 265-268, Sarajevo, Bosnia and Herzegovina, June 2011, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5977353

A Generalization of Quad-Trees Applied to Shape Coding

Rade Kutil and Christine Gfrerer
University of Salzburg, Department of Computer Sciences
Jakob Haringer-Str. 2, 5020 Salzburg, Austria
Email: rkutil@cosy.sbg.ac.at

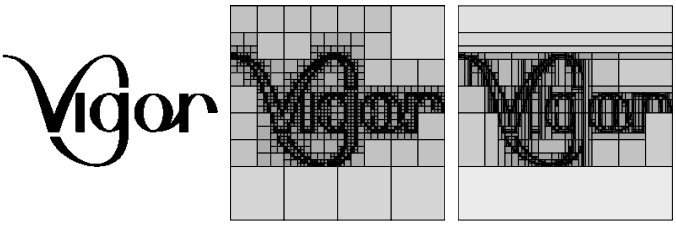
***Abstract*—Quad-trees are restricted to combined horizontal and vertical decomposition of tiles. If this restriction is dropped, anisotropic rectangular tiles result which can be arranged in a previously developed graph structure called "bush". A new tiling algorithm is able to reduce the number of tiles by a factor of about two. Together with an existing efficient coding scheme for this graph, a lossless coding algorithm for bi-level and indexed color images is developed. It is compared to JBIG2 and PNG, and turns out to be efficient for situations with large uniform regions and high color payload.**

I. Introduction

Quad-trees have long been used in general image coding [1], for bi-level images [2], and video coding [3]. In quad-tree coding, square tiles are recursively decomposed into four square sub-tiles. The process stops for sufficiently uniform tiles, for which the color payload is encoded. For bi-level or indexed color images, this payload is the pixel color. For natural images, image segments are approximated by planar [4] or polynomial [5], [6] functions. Generalized tilings with arbitrarily oriented linear splitting are used here, though, to achieve a more accurate approximation of region borders. However, this leads to increased bit budgets for encoding of the segmentation structure, so block merge algorithms [7] or combinations with quad-trees [8] have been developed.

These schemes can be applied to DCT [9] and wavelet [10] coding, as well as motion estimation [11]–[13] in video coding. Common to these applications is the amount of data to be encoded per tile, which is larger than the single bit payload for bi-level images, where JBIG2 [14] or chain codes [15] are far superior. Another reason to use tree structures to encode image data is the ability to arbitrarily select spatial details, as needed in terrain visualization [16] and display of geospatial data [17]. Also, spatial databases use quad-trees [18] in a similar way.

This work exploits a generalization of quad-trees that retains a redundancy-free representation. Tiles may be split anisotropically in horizontal or vertical dimension, which may produce highly non-square tiles. The number of these decompositions was shown to be much higher than that of quad-trees [19], [20]. Moreover, the representation as a binary tree of horizontal or vertical splits is not unique, and, therefore, causes redundancy and inefficiency in coding. However, if the graph structure is expanded to incorporate all possible decomposition trees with the same set of leaf nodes, uniqueness is achieved.



(a) shape (b) quad-tree (c) bush

Fig. 1. A shape and its decomposition into 3514 quad-tree tiles and 1861 bush tiles.

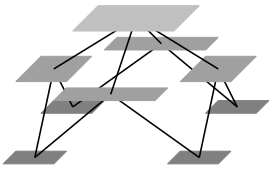
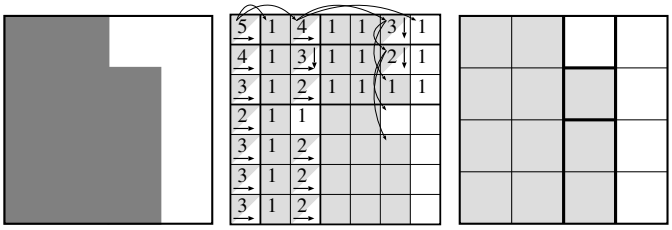


Fig. 2. A full bush of anisotropic tiles

In [20]–[22], such a graph, called "bush", together with an efficient redundancy-free coding algorithm has been developed. Redundancy-free means that there is only one representation for each set of tilings and, when encoded, no encoded symbol can be deduced from other parts of encoded data. In this work, an algorithm to find the best bush-tiling for an image, in terms of tile count, is presented, and a coding scheme for the color information is developed. A class of images is generated by randomly growing blots, optionally followed by smoothing operations, to test the coding efficiency and to compare it to JBIG2 and PNG.

II. Tiling Algorithm

Anisotropic tilings lead to non-unique decompositions in terms of binary trees. See e.g. Fig. 2, where four square sub-tiles can be produced either by horizontal followed by vertical splitting, as well as in the reverse order. The coding of such a decomposition tree is, therefore, necessarily redundant. By incorporating all decomposition trees into a two-dimensional graph structure, called "bush", a unique representation is achieved. This graph is basically a Cartesian product of two binary trees, or a subgraph thereof. It is not a tree because it contains cycles. The condition that the bush must contain all possible decomposition trees can be checked and enforced in



(a) shape (b) algorithm (c) result

Fig. 3. Example for optimal tiling algorithm

(a) raw (b) smoothed (c) aligned (d) colors

Fig. 4. Generated shapes made of 40 blots. (a) has 3.9% border pixels, (b) is smoothed with a 33×33-filter to 2.4% border pixels, (c) is aligned with a 65×5-filter to 1.17% aligned and 1.07% diagonal border pixels, (d) has 16 colors with one blot for each color.

a local fashion by inheriting complete splits up and down the hierarchy of nodes. This allows to encode splits as early as possible in a top-down coding scheme and to omit redundant split information at deeper levels in the graph. For an efficient coding algorithm of bushes, see [20]–[22].

However, there are still several possible bush tilings for a given shape, contrary to quad-trees, which are always unique. As these bush tilings have different numbers of tiles, we want find the one with the lowest number of tiles. See Fig. 1 for an example. First, we need a way to arrange all possible tiles of an image. Since a bush is a product of two binary trees, and a binary tree can be arranged as a 1-D array, bush-nodes can be arranged in a 2-D array of size $2m-1 \times 2n-1$, where $m \times n$ is the image size, and m and n are, for simplicity, a power of two. See Fig. 3 for an example. The right lower $m \times n$-block of this array is filled with image data. Each node is associated with the best number of tilings it can be decomposed into. This number is calculated from the lower right to the upper left corner. If a tile is uniform, it gets a 1. Otherwise, it gets the minimum of the sum of the horizontal or vertical sub-tiles, and is associated with the corresponding decomposition dimension. In the end, the upper left node contains the minimum number of tiles for the image. Following the optimal decomposition dimensions, starting from the upper left node, the optimal bush can be created.

Image sizes that are not a power of two can be handled easiest by simply expanding the image to power-of-two size and filling the expanded area with the color of nearest margin pixels. In this way, only a slight enlargement of the bush and only a few additional tiles are required. When decoding an image, the expanded area is discarded.

III. Color Coding

After the bush structure in encoded using the algorithm in [20]–[22], the color information has to be encoded for each leaf tile. For bi-level images, each leaf tile is encoded with one bit (to be more precise, one out of two symbols in the arithmetic coder), except if two sibling tiles are both leaves, in which case only one tile has to be encoded, the other one must have the other color. For quad-trees, a similar scheme is used. Only for a set of four sub-tiles that are all leaves, a second model is used in the arithmetic coder to encode all sub-tiles together as one out of 14 symbols, the two combinations of all equal colors are not possible.

For multi-color images, the situation is more complicated. Colors that appear at one point in a sub-tree or sub-bush are more probable than other colors to appear in other nodes in the same sub-tree or sub-bush. Therefore, when a leaf-node is encoded, its color is passed on to its parent node and to all child nodes of the latter. Before encoding a leaf-node's color, the information is encoded whether its color is equal to the color passed on from its parent node. The color only has to be encoded if a "no" has been encoded, and the passed-on color is removed from the arithmetic coder's model. In the case of sibling leaf nodes, "no" can be assumed without coding for the second node (in the bush case) or the last node (in the quad-tree case when the first three nodes have equal color) because these nodes must have a color that is different from color that is passed on from the sibling nodes.

IV. Test Image Generation

To test the coding performance, we need a set of test images with a range of properties. They should contain areas of constant color with arbitrary borders, similar to geographical maps. This is done by randomly growing blots. For each color, a certain number of pixel seeds are placed on the image and stored in a buffer of border pixel positions. Buffer entries are randomly chosen, and, after coloring the corresponding pixel, their neighbors in four directions are inserted into the buffer. Pixels that are already colored are discarded. The process stops when the buffer is empty. See Fig. 4 (a) for an example.

Because the result has very ragged borders, an optional smoothing filter is applied. It takes the form of a block centered around each pixel. The pixel's color is substituted by the most frequent color in that block. The larger the block, the smoother the result will be. See Fig. 4 (b) for an example. To quantify the smoothness of the result, all (overlapping) 2×2-blocks of the image are classified as either mono-colored or, otherwise, as border blocks. The rate of border blocks approximates the rate of border pixels. It is determined by the number of blot seeds, the number of colors, and the smoothing block size. The bit-rate of compressed images is expected to grow linearly with this rate.

Moreover, anisotropic bush tilings prefer horizontally or vertically aligned borders for obvious reasons. To achieve such an alignment, the filter block is modified to have a cross shape made of two rectangular blocks of size $a \times b$ and $b \times a$ respectively. See Fig. 4 (c) for an example. To quantify also

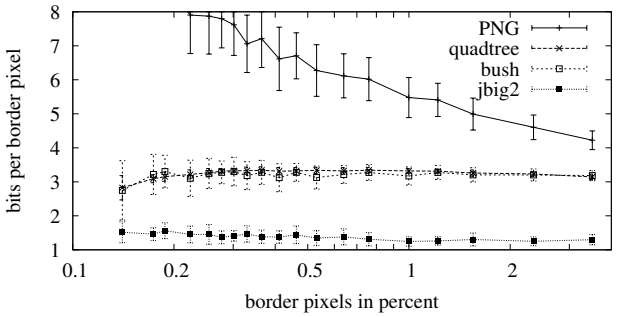


Fig. 5. Bit-rate depending on the number of pixels lying at region borders

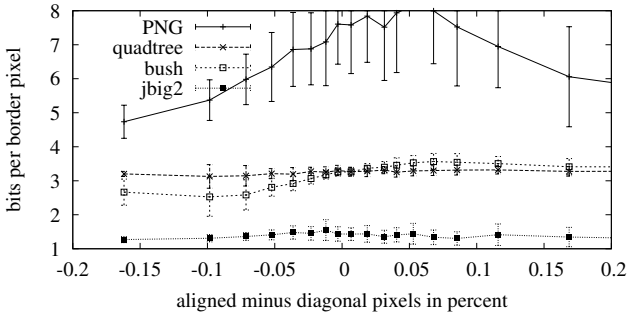


Fig. 6. Bit-rate depending on the degree of alignment (horizontal and vertical) of region borders

the alignment of blot borders, we further classify border blocks into diagonal blocks if two diagonal pixels are equal, and aligned blocks otherwise.

V. Experimental Results

A total of 745 bi-level images and 1292 colored images of size 1024×1024 have been generated to test the performance of our quad-tree and bush coding schemes. Fig. 5 shows overall results depending on the rate of border pixels for bi-level images. As the bit-rate is expected to grow linearly with this rate, it is not calculated in bits per image pixel but in bits per border pixel, so that a constant curve indicates a linearly growing bit-rate. Individual results are grouped into bins of equal number of images, and the average bit-rate plus-minus the standard deviation is shown. JBIG2 is superior and PNG is inferior compared to our coding schemes. Note that all schemes exhibit a linear growth of bit-rate with the border pixel rate, except for PNG, which cannot benefit as much from larger blots of constant color.

Quad-tree and bush tilings show approximately equal performance for general bi-level images. However, bush tilings have much larger deviation. This indicates that there are some images where bush tilings perform significantly better, and others where they are worse. It turns out that the border alignment is what causes this phenomenon. Fig. 6 shows that the bit-rate drops by almost one bit per border pixel for images with more aligned than diagonal pixels. This is of importance because many images naturally incorporate horizontal and vertical features. The reason for this behavior is the reduced

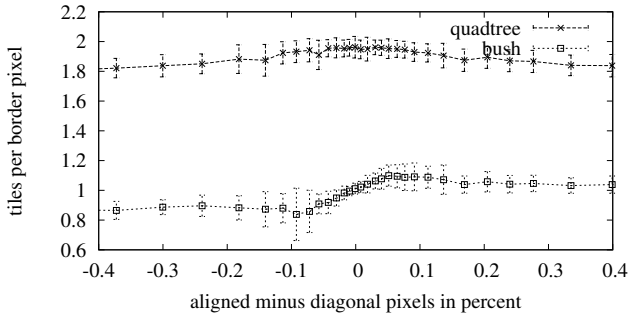


Fig. 7. Number of tiles depending on the degree of alignment of region borders

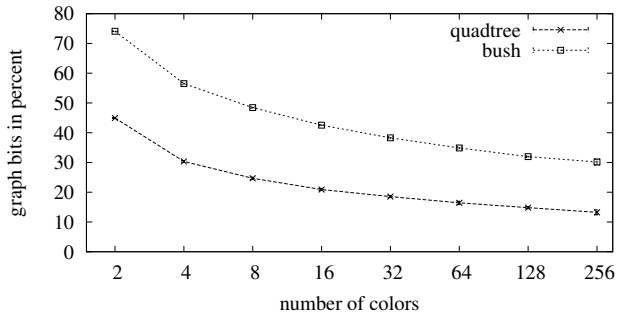


Fig. 8. Share of bits used to encode the graph structure

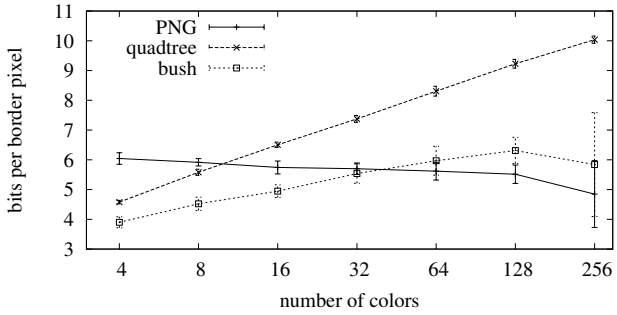


Fig. 9. Bit-rate depending on the payload, i.e. number of colors, for border pixel rates between 3 and 4

number of anisotropic tiles for such images, as can be seen in Fig. 7.

This figure also shows that bush tilings are able to reduce the number of tiles by approximately two for all images. Accordingly, the bit-rate used for encoding the graph structure is much higher for bush tilings than for quad-tree tilings, as can be seen in Fig. 8. Conversely, the bit-rate for color information is much lower, due to the reduced number of tiles. This effect even grows with the number of colors because the bit-rate for color coding becomes more prominent.

As a result, a higher number of colors increases the competitiveness of bush tilings, as can be seen in Fig. 9. Bush tilings are clearly superior to quad-trees, and the quotient of their bit-rates evolves in favor of bush tilings while the number of colors grows. Note that the border pixel rate is quite high

in Fig. 9 in order to fit 256 blots into the image. This is the reason that PNG is able to beat bush tilings for images with over 40 colors. For larger images, blots are also larger, and PNG performs far worse, as shown in Fig. 5. Note also that the bit-rate for PNG remains about constant in Fig. 9 while that of quad-tree and bush tilings increases linearly. This shows that the bit budget for representing shapes is prominent in PNG while the pure color information is comparably neglectable.

VI. Conclusions

Anisotropic tilings, when applying a new algorithm for optimal tiling, are able to represent a shape with only half the number of tiles compared to quad-tree tilings. However, there are much more possible tilings and the tiling structure is a more complicated graph, a so-called "bush". Therefore, a bigger part of the bit-rate has to be devoted to encoding the tiling. Nevertheless, the reduced number of tiles reduces the bit-rate for the payload, i.e. the tile color information, so that the overall bit-rate is improved especially for high payload, i.e. high numbers of colors.

As the bit-rate depends linearly on the number of pixels at the border of blots of uniform color, contrary to schemes such as PNG, where the bit-rate depends on the image size, bush tilings are suitable for images with large mono-colored blots. Such images may be found in geographical maps. In those applications, the ability to arbitrarily select spatial details is important, a feature that is carried over from quad-trees to bushes. Moreover, bush tilings prefer horizontally and vertically aligned blot borders which are common in technical diagrams as well as images of artificial objects.

Bush tilings can also be applied to natural image and video coding in the same way as quad-trees are used. Here, the tile payload consists of parameters for planar or polynomial approximations of image content or motion vectors in the case of motion compensated video coding. Such a rather big payload, compared to indexed color images, makes bush tilings especially interesting. Rate-distortion curves may be shifted to produce better image approximations with more detailed and better fitting tilings with the same or lower bit budget for tile payload.

Bush tilings can be seen as a compromise between quad-trees and general segmentations with arbitrarily oriented linear splittings. They offer more flexibility and adaptability than quad-trees while retaining a redundancy-free representation, which is important for efficient coding.

References

[1] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," *IEEE Transactions on Image Processing*, vol. 3, no. 3, pp. 327–331, May 1994.

[2] M. Manohar, P. S. Rao, and S. S. Iyengar, "Template quadtrees for representing region and line data present in binary images," *Computer Vision, Graphics, and Image Processing*, vol. 51, no. 3, pp. 338–354, 1990.

[3] M. Lightstone and S. K. Mitra, "Quadtree optimization for image and video coding," *Journal of VLSI Signal Processing*, vol. 17, pp. 215–224, 1997.

[4] M. Sarkis and K. Diepold, "Content adaptive mesh representation of images using binary space partitions," *IEEE Transactions on Image Processing*, vol. 18, no. 5, pp. 1069–1079, May 2009.

[5] R. Shukla, P. L. Dragotti, M. N. Do, and M. Vetterli, "Rate-distortion optimized tree-structured compression algorithms for piecewise polynomial images," *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 343–359, Mar. 2005.

[6] H. Radha, M. Vetterli, and R. Leonardi, "Image compression using binary space partitioning trees," *IEEE Transactions on Image Processing*, vol. 5, no. 12, pp. 1610–1624, Dec. 1996.

[7] C. S. Won, "A block-based MAP segmentation for image compressions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 592–601, Sep. 1998.

[8] A. A. Kassim, W. S. Lee, and D. Zonoobi, "Hierarchical segmentation-based image coding using hybrid quad-binary trees," *IEEE Transactions on Image Processing*, vol. 18, no. 6, pp. 1284–1291, Jun. 2009.

[9] K. Lengwehasatit and A. Ortega, "Rate-complexity-distortion optimization for quadtree-based DCT coding," in *Proceedings of the IEEE International Conference on Image Processing, ICIP 2000*, vol. 3, Sep. 2000, pp. 821–824.

[10] C. Y. Wang, S. J. Liao, and L. W. Chang, "Wavelet image coding using variable blocksize vector quantization with optimal quadtree segmentation," *Signal Processing: Image Communication*, vol. 15, no. 10, pp. 879–890, 2000.

[11] V. Argyriou and T. Vlachos, "Quad-tree motion estimation in the frequency domain using gradient correlation," *IEEE Transactions on Multimedia*, vol. 9, no. 6, pp. 1147–1154, Oct. 2007.

[12] J. Zhang, M. O. Ahmad, and M. N. S. Swamy, "Quadtree structured region-wise motion compensation for video compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 5, pp. 808–822, Aug. 1999.

[13] I. Rhee, G. R. Martin, S. Muthukrishnan, and R. A. Packwood, "Quadtree-structured variable-size block-matching motion estimation with minimal error," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 42–50, Feb. 2000.

[14] F. Ono, W. Rucklidge, R. Arps, and C. Constantinescu, "JBIG2 – the ultimate bi-level image coding standard," in *Proceedings of the IEEE International Conference on Image Processing, ICIP 2000*, vol. 1, Sep. 2000, pp. 140–143.

[15] H. Sánchez-Cruz, E. Bribiesca, and R. M. Rodríguez-Dagnino, "Efficiency of chain codes to represent binary objects," *Pattern Recognition*, vol. 40, no. 6, pp. 1660–1674, 2007.

[16] K. Baumann, J. Döllner, K. Hinrichs, and O. Kersting, "A hybrid, hierarchical data structure for real-time terrain visualization," in *Proceedings of the Computer Graphics International conference, CGI 1999*, 1999, pp. 85–92.

[17] J. Zhang and S. You, "Supporting web-based visual exploration of large-scale raster geospatial data using binned min-max quadtree," in *Proceedings of the 22nd international conference on scientific and statistical database management*, ser. SSDBM'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 379–396.

[18] R. K. Kothuri, S. Ravada, and D. Abugov, "Quadtree and R-tree indexes in Oracle spatial: a comparison using GIS data," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '02. New York, NY, USA: ACM, 2002, pp. 546–557.

[19] D. Xu and M. N. Do, "On the number of rectangular tilings," *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 3225–3230, Oct. 2006.

[20] R. Kutil and D. Engel, "Methods for the anisotropic wavelet packet transform," *Applied and Computational Harmonic Analysis*, vol. 25, no. 3, pp. 295–314, 2008.

[21] R. Kutil, "The graph structure of the anisotropic wavelet packet transform," in *Proceedings of the 7th international scientific conference devoted to the 25th anniversary of civil engineering faculty and 50th anniversary of technical university Kosice*, May 2002, pp. 41–47.

[22] ——, "Wavelet domain based techniques for video coding," Ph.D. dissertation, Department of Scientific Computing, University of Salzburg, Austria, Jul. 2002.