EDGE-PRESERVING WAVELET IMAGE REPRESENTATIONS WITH APPLICATIONS TO COMPRESSION OF AERIAL ORTHO IMAGES

Armein Langi[1]

[1] School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, Indonesia, Tel: +62-815-700-5930, Fax: +62-22-250-0985, e-mail: langi@dsp.itb.ac.id

Received Date: July 12, 2011

Abstract

This paper describes a new second-generation-type lossy image compression scheme using wavelets for aerial ortho images. For map applications where some losses are allowed, aerial ortho images present a possibility of having a very high compression ratio, because they are natural images with many uniform (coherent) subregions. However classical compression schemes such as the joint-photograph expert group (JPEG) standard do not preserve image edges, which are critical in preserving precision of the objects' positions. In our compression scheme, an image is seen as a collection of edges and textures, instead of pixels, hence edge preservation can be made explicit. This scheme is particularly promising because the images contain many man-made, regular objects, such as roads, houses, parks, and farmlands. The scheme uses wavelet extrema to extract the edges. The paper shows that an excellent reconstruction of aerial ortho images can be obtained from edges. The paper also discusses techniques to compress edges of the aerial ortho images.

Keywords: Aerial ortho images, Edge preservation, Multiscale edge detection, Wavelets

Introduction

Aerial ortho images provide digital land-related information that is very useful for regional development and utility planning. The images are generated by computers from aerial photographs through geodetic transformations, compensating acquisition (movement) distortions and the earth curvature. As a result, the aerial ortho images feature high precision of pixel positions and very flat landscape images, suitable for land-related planning [1]. Unfortunately, such features also cause the image sizes to be large, typically 5000×5000 pixels in 8 bits per pixel (bpp) resulting in 25 Mbytes data per image [2]. Compression schemes are needed because otherwise such large images are difficult to manage (e.g., to store and distribute). Compression allows more images to be stored and reduces transmission time.

At a 7.65 bits/pixel entropy level, such images are difficult to compress losslessly, or near losslessly. In the past, we have used predictive coding, joint-photograph expert group (JPEG), and wavelet denoising schemes to compress aerial ortho images for various applications near losslessly [2]. Predictive coding models a pixel in the image as a linear combination of its adjacent pixels. One can then use the linear combination to predict the pixel. The error of the prediction is called residue. A lossless and near lossless scheme can then compress the residue as well as prediction coefficients. The image quality is very high (in the order of 51.1 dB peak signal-to-noise ratio, PSNR) but at a compression level of as low as 2:1. The JPEG coder uses discrete cosine transform, combined with block coding and entropy coding. At a 2.08:1 compression ratio, JPEG achieves 44.7 dB PSNR. A wavelet denoising scheme reduces the wavelet coefficients uniformly using various threshold levels. At a 2:03:1 compression ratio, wavelet denoising scheme performs rather poorly, 35.58 dB PSNR only.

Lossy compression of aerial ortho images can achieve high compression ratios because the images contain natural objects, such as houses, roads, driveways, parks, and farmlands, which are regular objects. Such regularity contains redundancy that increases compression ratio. Furthermore, many applications use the images as maps, requiring precision in edge position. Therefore many parts of textures can be considered irrelevant and can be removed to increase the compression ratio. Unfortunately classical schemes such as the JPEG standard do not preserve edge positions, which are critical features of aerial ortho images.

The paper proposes a new second-generation image coding for aerial ortho images based on wavelet transform to satisfy both requirements, i.e., high compression ratios and edge preservation. The coding is considered second generation because it treats images as collections of edges and textures instead of merely collections of pixels [3]. It is then possible to preserve the edges explicitly. Clearly the edge-preserving approach is only possible if there are ways to: (i) extract edges and textures, (ii) reconstruct the image from its edges and textures, (iii) represent edges and textures efficiently, and (iv) distinguish between significant and insignificant edges and textures. This paper shows how the proposed scheme provides all four requirements.

The main objective of this paper is to combine wavelet image representation schemes, image reconstructions, edge representation, edge quantization, and entropy coding into a working compression scheme for aerial ortho images, in a way that can be easily replicated and implemented for various geodetical and map applications. Each concept has been developed separately for different applications. This paper incorporates each concept into a compression scheme, and adjusts it for geodetical images.

The coding scheme is based on the works by Mallat and Zhong [4] and Cvetkovic and Vetterli [5] on edge and texture extraction and reconstruction using wavelet transforms, as well as the work by Carlsson [6] on efficient edge representation (sketch coding). The coding extracts edges by extracting multiscale extrema of wavelet coefficients. This approach allows identification of more meaningful edges than those extracted by the (single-scale) Laplacian operator used by Carlsson [6], as there are now extrema from more than one scales to be used in the detection. Furthermore, such edges can be used to reconstruct the images with excellent approximation quality. Both [4] and [5] provide convex iterations to approximate the missing wavelet coefficients between edges. An inverse wavelet transform of the resulting coefficients (including the edge coefficients) approximates the original image and preserves the edges and basic textures.

As shown in Figure 1, the compression scheme finds the multiscale edges using wavelet transform [7]. It represents the edges efficiently using the sketch coding [6]. If necessary (i.e., the basic textures generated from edges are not sufficient), the scheme extracts the texture image and compresses it using standard image compression. One simple way to extract the texture image is to reconstruct the best image from the edges, and subtract it from the original image. Finally, the scheme saves or transmits the compressed edges and/or texture data.

A decompression scheme (shown in Figure 2) recovers the edges and textures from the compressed data. Using the convex iteration, the scheme reconstructs the image from the edges. If needed, the scheme adds the texture image to the reconstructed image.

The next sections describe the steps in the schemes. This paper assumes that, for map applications, the basic textures generated from edges are sufficient. Section 2 explains the edges extraction using wavelets. Section 3 describes how to reconstruct an image from its wavelet edges. Section 4 explains the sketch coding to compress the edges. It also reports some preliminary results. Section 5 gives some conclusions.

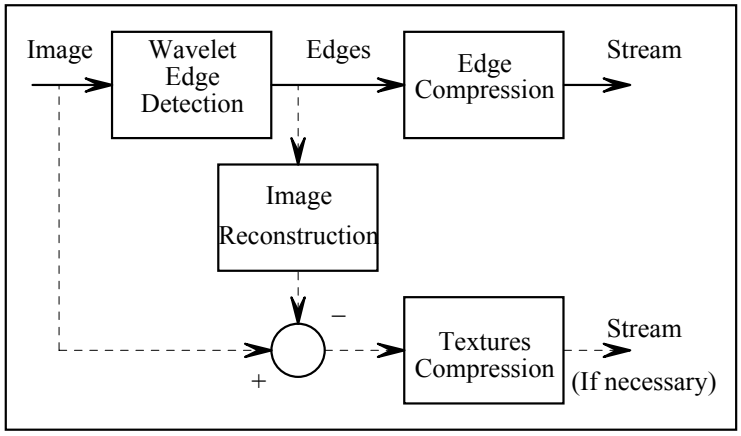


Figure 1. Edge-preserving compression scheme.

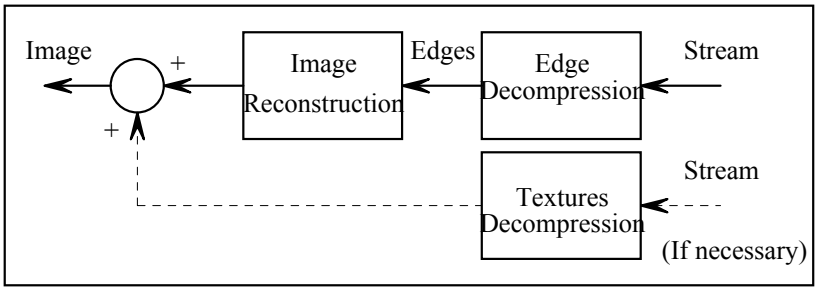


Figure 2. Edge-preserving decompression scheme.

Edges Extraction Using Wavelets

The primary task is to classify pixels into edge and texture pixels. Edges are usually associated with objects. However, algorithms to identify edges through object identification are generally difficult. A simpler solution is to observe sharp transitions of grayscale values of image pixels $f(x,y)$, where x and y are the pixel position coordinates at the horizontal and vertical directions, respectively. Edges are then redefined as collections of pixels where the transitions occur. Problems occur when a transition does not come from an object edge or when object edges do not translate into sharp transitions. In both cases the algorithm produces "false edges" (i.e., edges that do not correspond to physical edges) and/or misses "true edges" (i.e., edges that correspond to physical edges). However, for aerial ortho images, the probability of making such mistakes can be considered small because they contain many regular objects. Furthermore, the penalty for acquiring many false edges is not in the image quality, but in the compression ratio.

It can be shown that the wavelet transforms result in extrema at the grayscale value transitions, hence wavelet edge detection algorithm is essentially an extrema detection [4]. Consider two-dimensional (2D) wavelet signals $\psi_s^H(x,y)$ and $\psi_s^V(x,y)$ for horizontal and vertical directions, respectively, with $s \in R$ is a *scaling* factor. The *continuous wavelet transform* (CWT) of $f(x,y)$ is defined using convolutions, $*$, as

$$W_s^H f(x,y) = f(x,y) * \psi_s^H(x,y)$$
$$W_s^V f(x,y) = f(x,y) * \psi_s^V(x,y) \quad (1)$$

In this work, we have selected a *dyadic* sequence $(2^j)_{j \in Z}$ for s, resulting in a *dyadic* CWT. Dyadic CWT allows a more efficient, stable representation and a faster *pyramidal*

algorithm. There now exists *reconstructing* wavelet functions $\chi_{2^j}^H(x,y)$ and $\chi_{2^j}^V(x,y)$ such that the inverse CWT is

$$f(x,y)=W^{-1}\left(W_{2^j}^H f, W_{2^j}^V f\right)_{j\in Z}=\sum_{j\in Z}\left(W_{2^j}^H f * \chi_{2^j}^H(x,y)+W_{2^j}^V f * \chi_{2^j}^V(x,y)\right) \tag{2}$$

In practice, signal $f(x,y)$ is bandlimited, hence the scale exponent j can take positive integers only. Furthermore, if we limit j to be $1 \le j \le J$ with a user-selected J, we have

$$f(x,y)=S_{2^J} f+\sum_{j=1}^{J}\left(W_{2^j}^H f * \chi_{2^j}^H(x,y)+W_{2^j}^V f * \chi_{2^j}^V(x,y)\right) \tag{3}$$

where $S_{2^J} f(x,y)$ represents all the remaining terms for $j > J$.

Let us construct a complex wavelet transform of an image $f(x,y)$ from its wavelet representation as $W_{2^j}^H f(x,y)+iW_{2^j}^V f(x,y)$ such that we can define wavelet *modulus* of $f(x,y)$

$$M_{2^j} f(x,y) = \sqrt{\left|W_{2^j}^H f(x,y)\right|^2 + \left|W_{2^j}^V f(x,y)\right|^2} \tag{4}$$

and wavelet *angle* of $f(x,y)$ as

$$A_{2^j} f(x,y)=\arg\left(W_{2^j}^H f(x,y)+iW_{2^j}^V f(x,y)\right) \tag{5}$$

The modulus-angle representation using Eqs. (4) and (5) is equivalent with the wavelet representation in Equation (1) because one can always obtain the wavelet coefficients from the modulus and angle data, according to

$$W_{2^j}^H f(x,y)=M_{2^j} f(x,y)\cos\left(A_{2^j} f(x,y)\right) \tag{6}$$

and

$$W_{2^j}^V f(x,y)=M_{2^j} f(x,y)\sin\left(A_{2^j} f(x,y)\right) \tag{7}$$

This is an analogy to the relationship between vectors and their polar representations. It should be noted that this modulus-angle representation can code any edges completely, including diagonal edges. Diagonal edges are recognized from their angles in Equation (5). However, if all edges are diagonal of a certain direction, a diagonal wavelet representation may be simpler and more efficient.

The edges can now be obtained through *extrema* of the modulus (called *modulus extrema*). The modulus extrema are all the $\left(M_{2^j} f(x,y), A_{2^j} f(x,y)\right)$ at points (x,y) having modulus that are larger than moduli of their two neighboring points whose positions are in the direction indicated by the angle value $A_{2^j} f(x,y)$. The modulus extrema (called $\left(M_{2^j}^e f, A_{2^j}^e f\right)$) correspond to the object edges. Hence, the connected modulus extrema pixels are *chained* to be the detected edges. As a result, the edges are chains of extrema that are connected positionally. If there are more than one extrema being in the neighborhood, the one that has the closest angle is selected in the chain.

Reconstruction Using Edges

Edge-Preserving Reconstruction

Suppose now we have the extrema $\left(M_{2^j}^e f, A_{2^j}^e f\right)$, how do we reconstruct the $f(x,y)$? Although perfect reconstruction is not attainable, an excellent approximation is possible. An iteration procedure described in [4] and [5] (shown in Figure 4) generates modulus-angle coefficients $\left(\widetilde{M}_{2^j} f(x,y), \widetilde{A}_{2^j} f(x,y)\right)$ for all (x,y) such that the extrema of them are

exactly $\left(M_{2^j}^e f, A_{2^j}^e f\right)$. Once the generated coefficients are available, Equation (6), (7) and then (3) can reconstruct an image $\tilde{f}(x,y)$. Clearly the $\left(\tilde{M}_{2^j} f(x,y), \tilde{A}_{2^j} f(x,y)\right)$ defined this way is not unique. However, not only the resulting $\tilde{f}(x,y)$ preserves the edges, but also basic textures are recovered, e.g., brighter textures in the original are brighter in the reconstructed image.

Iteration Projection

Let us start with defining our working modulus-angle coefficients $\left(\tilde{M}_{2^j} f(x,y), \tilde{A}_{2^j} f(x,y)\right)$ as zero everywhere except at edge points where they have values $\left(M_{2^j}^e f, A_{2^j}^e f\right)$. Using Equation (6) and (7) on the coefficients, we obtain $\tilde{W}_{2^j}^H f(x,y)$ and $\tilde{W}_{2^j}^V f(x,y)$ which are also zero everywhere except at edge points $\left(M_{2^j}^e f, A_{2^j}^e f\right)$, illustrated in the upper graph of Figure 3 where vertical lines represent wavelet coefficients. Call the coefficients at edge points as $G_{2^j}^H(x,y)$ and $G_{2^j}^V(x,y)$. The iteration steps shown in Figure 4 are then

(1) Inverse CWT $\tilde{W}_{2^j}^H f(x,y)$ and $\tilde{W}_{2^j}^V f(x,y)$ into an image $\tilde{f}(x,y)$ using Equation (3).

(2) Forward CWT the image $\tilde{f}(x,y)$ back to new $\tilde{W}_{2^j}^H f(x,y)$ and $\tilde{W}_{2^j}^V f(x,y)$ using Equation (1). This ensures that the new coefficients are in the range of the forward CWT.

(3) Adjust $\tilde{W}_{2^j}^H f(x,y) = G_{2^j}^H(x,y)$ and $\tilde{W}_{2^j}^V f(x,y) = G_{2^j}^V(x,y)$ for (x,y) in the edges by explicitly replacing their values. This ensures that the new coefficients have the same values at the edges as those in the original, as shown in the middle graph of Figure 3.

(4) Adjust $\tilde{W}_{2^j}^H f(x,y)$ and $\tilde{W}_{2^j}^V f(x,y)$ between two edges to preserve monotonicity of $\tilde{W}_{2^j}^H f(x,y)$ and $\tilde{W}_{2^j}^V f(x,y)$ in those edges. For example, if the wavelet coefficient of one edge is larger that of the next edge, the coefficients in between must be monotonically decreasing, illustrated in bottom graph of Figure 3. If two consecutive coefficients violate the monotonicity, both are replaced by their arithmetic mean. This ensures that the extrema are preserved.

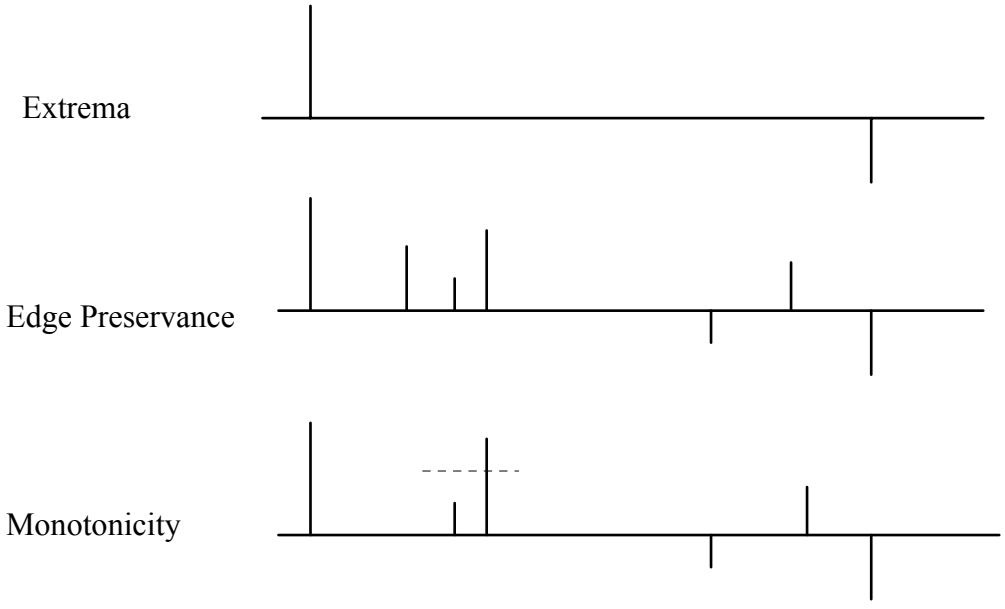


Figure 3. Adjustments to preserve wavelet coefficient monotonicity.

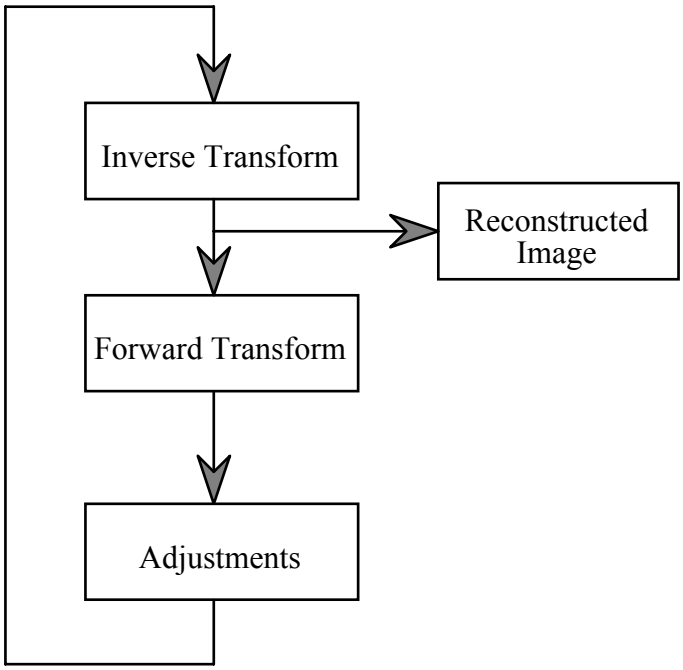


Figure 4. An iterative image reconstruction from edges.

It has been shown in [5] and [7] that the steps in Figure 4 represent a convex iteration, which converges to a set of $\tilde{W}_{2^j}^H f(x,y)$ and $\tilde{W}_{2^j}^V f(x,y)$ satisfying two requirements: the extrema are preserved and the set is in the range of CWT. This means the corresponding $\tilde{f}(x,y)$ is in the same image domain as $f(x,y)$ and both have the same object edges.

Figure 5 shows the effect of the iteration process on the working coefficients. The upper left is the working original image. The upper right is an image containing the edges. It is a working image at a scale $j = 1$. The lower left is the reconstructed image without any iteration, and the lower right is the reconstructed image after 10 iterations. The reconstructed image is almost perceptually indistinguishable from the original at a PSNR of 35 dB.

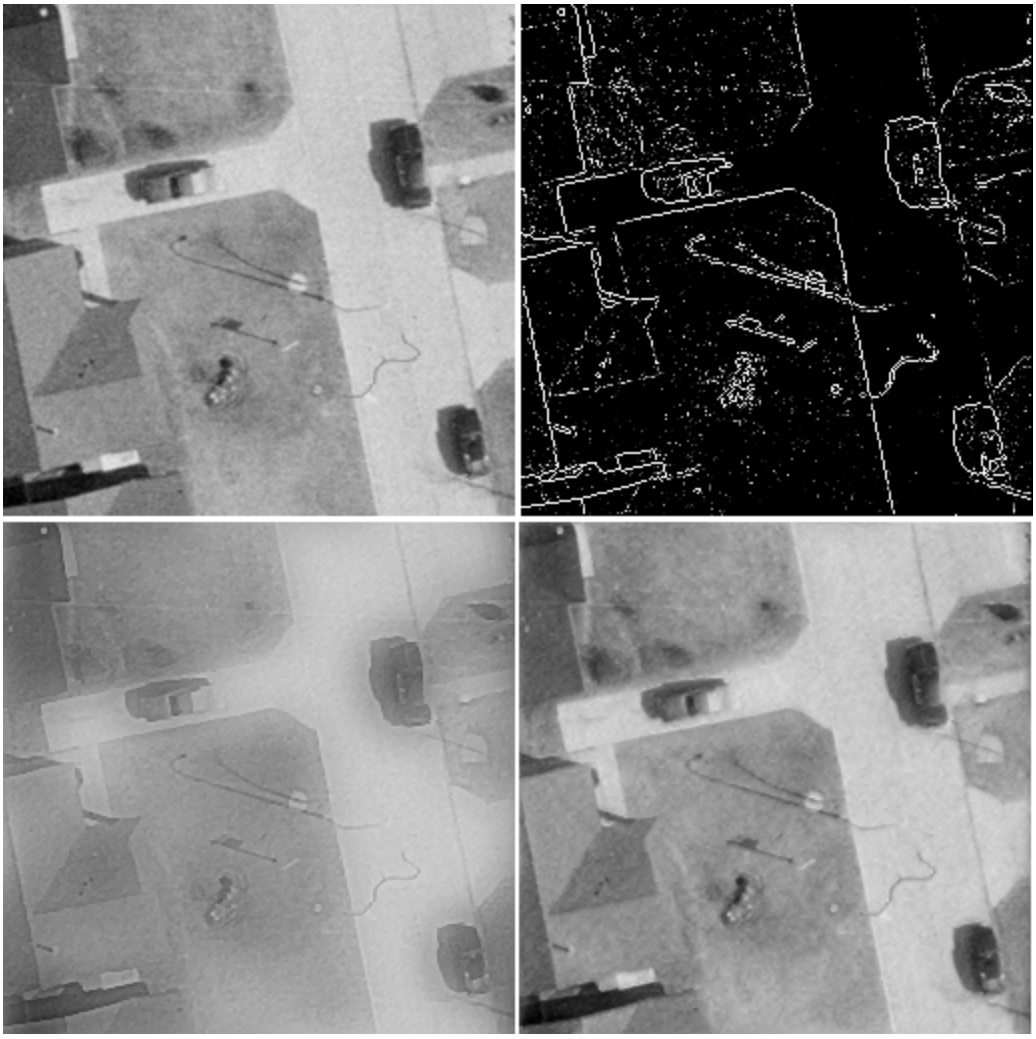


Figure 5. Image reconstruction from edges: (upper left) the original image. (upper right) one edge image at one wavelet scale. (lower left) the reconstructed image before iteration. (lower right) the reconstructed image after 10 iteration cycles.

Compression of Edges

Edge Coding

As they are, the edge chains still use too many bits for image representation. There are three aspects of points in an edge chain that must be preserved: geometry (i.e., (x,y) positions), the modulus, and the angle. The primary compression parameter (i.e., the parameter that governs the quality vs. bit rate performance) is the number of edge chains being transmitted. The more edge chains are sent, the better quality of the reconstructed image is. However, that means more bits are required to send the points. We then need ways to determine which edge chains are more important than the others.

One way to find important edge chains is based on the fact that edges of natural objects usually have extrema at the same positions (x,y) at different scales j. Hence, an algorithm checks whether a chain in a scale has corresponding chains at the same position at different scales. If yes, the chain is important, otherwise it can be ignored. Another way is based on the chain length (i.e., the number of points in a chain). Important edges tend to have long

chains. Thus, one can set a length threshold, below which the edge chains are omitted. One other way utilizes the fact that a high modulus value signifies a sharp transition of grayscale values, hence it is more important than low modulus values. One can then set thresholds for the modulus values of the individual points or their sum in a chain. Chains having such values below the thresholds are ignored. Once the edge chains have been selected, they are coded by coding the edge points (i.e., the geometry, modulus, and angle of each point).

Chain Geometry Coding

A simple geometry coding creates a list of pixels position for every edge chain. The first entry is the coordinate of the first (starting) point. The actual data that are saved are its relative coordinate values with respect to the starting point of the previous edge. The next point position is one of eight possible neighbors: east (E), north-east (NE), north (N), north-west (NW), west (W), south-west (SW), south (S), and south-east (SE). Hence instead of having to transmit the actual coordinates (i.e., two integers for x and y), the scheme can transmit one of those eight symbols only (i.e., three bits). Furthermore, by assigning integer symbols to those eight neighbors, one can represent the next position by the difference of the next position symbol to that of the current position. Many edges have relatively smooth trajectories. Hence, the symbol differences are mostly small values, further reducing the entropy.

Chain Magnitude Coding

The magnitude coding utilizes the fact that the modulus values of edge points in a chain do not change in a random fashion. One simplest coding (called *delta* coding) is to save the first magnitude value in its actual value and then take the differences between the next values and the current values for the next pixels. The differences are scalar quantized to further reduce the bit requirements. A more complex coding uses predictions and second order polynomial approximations [6], resulting in 18 bits per edge chain, regardless the chain length. Mallat and Zhong [4] suggest a simple predictive coding using a coarse quantization of the prediction values. Furthermore, the chains are often sampled to further reduce the bit requirement.

Chain Angle Coding

In most cases the angle $A_{2^j} f(x,y)$ is not transmitted, because it can be inferred from geometric data, where $A_{2^j} f(x,y) = \pi/2 - \alpha$ relates the trajectory angle α and $A_{2^j} f(x,y)$. The angle α of the current point corresponds directly to the direction angle to its next point in the chain, which takes one of the eight angle values. Thus, Equation (8) provides a quantized approximation of the $A_{2^j} f(x,y)$.

A Coding Example

Figure 6 shows one example of edge-preserving coding schemes. The forward CWT and vectorial-to-polar conversion performs Equation (1) and (3)-(4), respectively. The extrema extraction obtains all possible edge points. Edge chaining selects important edges for a given set of length and magnitude thresholds. Finally, the compression performs the geometry and modulus coding. The compression module includes entropy coding such as arithmetic coding to compress the resulting geometry and modulus codes.

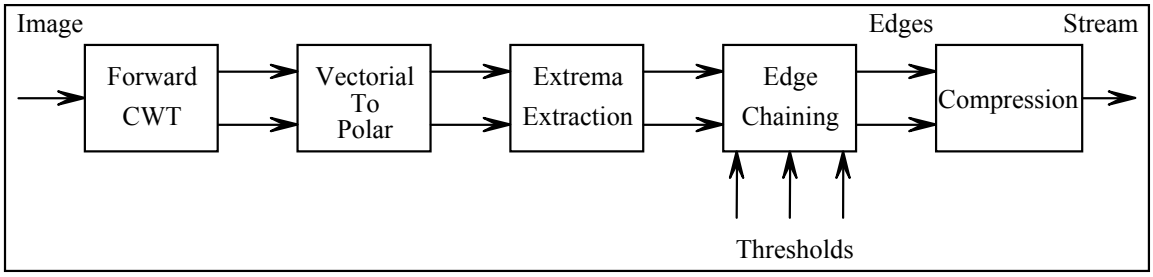


Figure 6. A practical scheme of edge-preserving compression.

The decompression scheme in Figure 7 recovers the edge chains, and converts the chains into $\left(\widetilde{M}_{2^j} f(x,y), \widetilde{A}_{2^j} f(x,y)\right)$ as a set of working coefficient set for the iteration. The iteration itself includes the forward and inverse CWT as explained in Section 3. After a user-defined number of iteration cycles, the scheme produces the desired image.

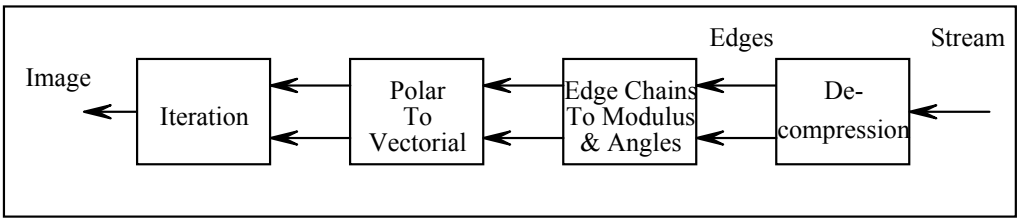


Figure 7. A practical scheme of edge-preserving de-compression.

It should be noted that the main advantage of the scheme is in its ability to preserve edges, but not textures. This scheme is particularly suitable for images containing important lines and edges. This compression scheme can be used on such cases including fingerprints, geodetical images, maps, land related pictures, and aerial ortho images. In any lossy coding, there will always be parts of the images that will be sacrificed for reduced number of bits. This scheme sacrifices texture information in preference to edges.

As a result, there are cases where a good edge preservation does not reflect in good PSNR. Figure 8 (a) shows an example of compression results. The modulus compression uses the delta coding. The image has 27.89 dB PSNR at about 13.8:1 compression ratio. Notice that there is no blocky effect. Furthermore, the basic textures are still recovered despite being estimated from the edge information only. As a comparison in Figure 7 (b), a JPEG scheme can achieve a compression ratio of 15:1 for approximately 30 dB peak signal to noise ratio (PSNR), but the edges become blurred, thus compromising the position precision.

This suggests a need for novel criteria other than PSNR to evaluate quality of compression of geodetic images. The criteria should provide more sensitivity toward valuable edge preservation. Texture imprecision may be tolerated, although must be within a limit where areas are still recognizable as what they are.

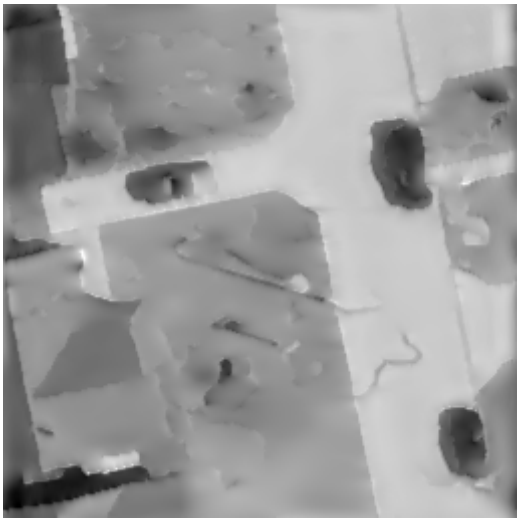
(a)  (b) 

Figure 8. (a) An example of compression results using the scheme in Figures 6 and 7, as compared to (b) a JPEG compressed image.

Conclusions

Compression on images of natural objects such as aerial ortho images allows high compression ratio at high values of PSNR. However it is important to add a quality requirement that the edges of the image objects are preserved to keep the high precision of object positions intact. Second generation compression schemes are suitable because they describe images in terms of edges and textures, hence the edge preservation can be made explicit. We have shown that such schemes are practical using wavelet-based algorithms for edge detection, edge coding, and image reconstruction from edges. The reconstruction algorithm approximates the texture through a convex iteration. As a result, the reconstructed image is of excellent quality. Not only the algorithms preserve the edges, but also they allow an approximation of basic textures. The challenge now is to compress the edges with as few bits as possible. The paper discusses the framework for edge compression and shows a preliminary result of aerial ortho image compression at 13.8:1 where important edges are preserved.

References

[1] R. Oswald, "Manitoba land-related information system: The information utility," In: *IEEE Wescanex 95 Proceedings*, pp. 252-257, May 1995.

[2] A. Langi, and W. Kinsner, *Optimal compression of large aerial ortho images*, *DSP RTG Technical Reports*, July 2010.

[3] M. Kunt, A. Ikonompoulus, and M. Kocher, "Second-generation image-coding techniques," In: *Proceedings IEEE*, Vol. 73, No. 4, pp. 549-574, April 1985.

[4] S. Mallat, and S. Zhong, "Characterization of signals from multiscale edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 7, pp. 710-732, July 1992.

[5] Z. Cvetkovic, and M. Vetterli, "Discrete-time wavelet extrema representation: Design and consistent reconstruction," *IEEE Transactions on Signal Processing*, Vol. 43, No. 3, pp. 681-693, March 1995.

[6] S. Carlsson, "Sketch based coding of grey level images," *Signal Processing*, Vol. 15, pp. 83, 1988.

[7] A. Langi, "Reconstructions of signals from singularities," *DSP RTG Technical Reports*, August 2010.