Compression and progressive transmission of astronomical images

Richard L. White

Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD 21218
rlw@stsci.edu

Jeffrey W. Percival

University of Wisconsin, Space Astronomy Laboratory, 1150 University Ave., Madison, WI 53706
jwp@sal.wisc.edu

ABSTRACT

An image compression algorithm has been developed that is well-suited to astronomical images. The method has 3 steps: an intensity mapping to generate an image that has roughly constant noise in each pixel, an orthonormal wavelet transform, and quadtree coding of the bit-planes of the wavelet coefficients. The quadtree values may be further compressed by any standard compression technique, such as Huffman or arithmetic coding. If the 2-dimensional Haar transform is used, the calculations can be carried out using integer arithmetic, and the method can be used for both lossy and lossless compression. The Haar transform basis functions are well-suited to most astronomical images because they are highly localized. The performance of the algorithm using smoother, longer range wavelets is also shown; they can give slightly better lossy compression at the cost of an increase in artifacts around point sources, but they are not effective for lossless compression using this scheme.

This technique has also been used as the basis of a progressive image transmission system that can be used for either remote observing or access to remote image archives. After less than 1% of the data have been received, the image is visually similar to the original, so it is possible to assess the quality of images very quickly. If necessary, the entire compressed data set can be sent so that the original image is recovered exactly.

1. INTRODUCTION

Astronomical images consist largely of empty sky. Compression of such images can reduce the volume of data that it is necessary to store (an important consideration for large scale digital sky surveys) and can shorten the time required to transmit images (useful for remote observing or remote access to data archives.) Astronomical images can be extremely large, making the potential gains from image compression very important. For example, the Space Telescope Science Institute has digitized photographic plates covering the entire sky, generating 1500 images each having 14000×14000 16-bit pixels. Several astronomical groups are now constructing cameras with mosaics of large CCDs (each 2048×2048 or larger); these instruments will be used in projects that generate data at a rate exceeding 100 MBytes every 5 minutes for many years.

Astronomical images have some unusual characteristics that make most existing image compression techniques either ineffective or inapplicable. A typical image consists of a nearly flat background sprinkled with point sources and occasional extended sources. The images are often noisy, so that lossless compression does not work very well; furthermore, the images are usually subjected to stringent quantitative analysis, so any lossy compression method must be proven not to discard useful information, but must instead discard only the noise.

An example may make clear the difficulties of astronomical image compression. One of the simplest data compression techniques is run-length coding, in which runs of consecutive pixels having the same value are compressed by storing the pixel value and the repetition factor. This method is used in the standard compression scheme for facsimile transmissions. Unfortunately, it is quite ineffective for lossless compression of astronomical images because even though the sky is *nearly* constant, the noise in the sky ensures that only very short runs of equal pixels occur. The obvious way to make run-length coding more effective is to force the sky to be exactly constant by setting all pixels below a threshold (chosen to be just above the sky) to the mean sky value. However, then one has lost any information about objects close to the detection limit. One has also lost information about local variations in the sky brightness, which severely limits the accuracy of photometry and astrometry on faint objects. Worse, there may be extended, low surface brightness objects that are not detectable in a single pixel but that are easily detected when the image is smoothed over a number of pixels; such faint structures are irretrievably lost when the image is thresholded to improve compression.

This paper describes an image compression algorithm that is well-suited to astronomical images. The method has 3 steps: (1) an intensity mapping to generate an image that has roughly constant noise in each pixel, (2) an orthonormal wavelet transform, and (3) quadtree coding of the bit-planes of the wavelet coefficients. The quadtree values may be further compressed by any standard compression technique, such as Huffman or arithmetic coding. This method is much better than techniques that keep only the wavelet coefficients with the largest amplitudes.

If the 2-D Haar transform is used as the wavelet transform, the calculations can be carried out using integer arithmetic, and the method can be used for both lossy and lossless compression. The Haar transform basis function are well-suited to most astronomical images because they are highly localized, and it is possible to adjust the coefficients during decompression to reduce the blockiness that comes from using such functions. The performance of the algorithm using smoother, longer range wavelets is also shown; they can give slightly better lossy compression, but they are not effective for lossless compression using this scheme.

This method is being used by the Space Telescope Science Institute to compress digitized versions of the Palomar and ESO Sky Survey plates for distribution on CD-ROM. Images compressed to about 1.5 bits/pixel are equivalent to the original images under both visual inspection and quantitative analysis. Images compressed to 0.2 bits/pixel are still useful, though some of the faintest objects are lost at such high compression factors.

This technique has also been used as the basis of a progressive image transmission system that can be used for either remote observing or access to remote image archives. After less than 1% of the data have been received, the image is visually similar to the original, so it is possible to assess the quality of images very quickly. If necessary, the entire compressed data set can be sent so that the original image is recovered exactly. It is also possible to speed the transmission even further by transmitting first only enough information to construct a version of the image that has been binned in blocks of 2×2 pixels; this is a natural feature of wavelet-based schemes.

2. THE H-TRANSFORM

The 2-dimensional Haar transform[1] (also known as the H-transform or the S-transform) can be used as the basis of an effective compression method for astronomical images[2−5]. The H-transform is calculated for an image of size $2^N \times 2^N$ as follows:

- Divide the image up into blocks of 2×2 pixels. Call the 4 pixels in a block a_{00}, a_{10}, a_{01}, and a_{11}.
- For each block compute 4 coefficients:

$$\begin{aligned} h_0 &= (a_{11} + a_{10} + a_{01} + a_{00})/2 \\ h_x &= (a_{11} + a_{10} - a_{01} - a_{00})/2 \\ h_y &= (a_{11} - a_{10} + a_{01} - a_{00})/2 \\ h_c &= (a_{11} - a_{10} - a_{01} + a_{00})/2 \end{aligned} \qquad (1)$$

- Construct a $2^{N-1} \times 2^{N-1}$ image from the h_0 values for each 2×2 block. Divide that image up into 2×2 blocks and repeat the above calculation. Repeat this process N times, reducing the image in size by a factor of 2 at each step, until only one h_0 value remains.

This calculation can be easily inverted to recover the original image from its transform. The transform is exactly reversible using integer arithmetic if one is careful with the low-order bits of the coefficients[5]. It is straightforward to extend the definition of the transform so that it can be computed for non-square images that do not have sides that are powers of 2; the most effective way to do this is to assume reflected boundary conditions at the edges of the image. The H-transform can be performed in place in memory and is very fast to compute, requiring about $16M^2/3$ (integer) additions for a $M \times M$ image.

The H-transform can be derived from the 1-dimensional Haar transform, which involves taking sums and differences of pairs of adjacent elements in a vector. Apply a single sum/difference step of the 1-D transform along the rows of the images, then along the columns of the transformed image. Repeat this row/column transform, using only the sum coefficients (1/4 of the original image) as input. Repeat until only a single element remains.

2.1. Other wavelet transforms

The H-transform is a simple 2-dimensional discrete wavelet transform. The compression scheme described here is easily adapted for use with other wavelet transforms. Any 1-dimensional discrete wavelet transform can be converted to a 2-D transform as outlined in the last section, and the coefficients of that 2-D transform can be efficiently coded using the schemes described below. In this paper, compression results are also shown for an algorithm based on the Daubechies D4 wavelet transform[6]. The D4 transform has been modified so that it uses reflected boundary conditions rather than periodic boundary conditions.

The major advantage of the H-transform over the Daubechies and similar wavelet transforms is that the H-transform can be performed entirely with integer arithmetic, making it exactly reversible. Consequently it can be used for either lossless or lossy compression (as indicated below) and one does not need a special technique for the case of lossless compression (as was required, *e.g.*, , for the JPEG compression standard and by FITSPRESS[7].) However, the smoothness afforded by higher-order transforms can be advantageous.

3. QUANTIZATION

If the image is nearly noiseless, the H-transform is somewhat easier to compress than the original image because the differences of adjacent pixels (as computed in the H-transform) tend to be smaller than the original pixel values for smooth images. Consequently fewer bits are required to store the values of the H-transform coefficients than are required for the original image. For very smooth images the pixel values may be constant over large regions, leading to transform coefficients that are zero over large areas.

Noisy images still do not compress well when transformed, though. Suppose there is noise σ in each pixel of the original image. Then from propagation of errors, the noise in each of the H-transform coefficients is also σ. To compress noisy images, divide each coefficient by $S\sigma$, where $S \sim 1$ is chosen according to how much loss is acceptable. This reduces the noise in the transform to $0.5/S$ (because the largest error is 1/2 the least significant bit of the quotient), so that large portions of the transform are zero (or nearly zero) and the transform is highly compressible.

Why is this better than simply quantizing the original image? As discussed above, if we divide the image by σ then we lose all information on objects that are within 0.5σ of sky in a *single* pixel, but that are detectable by averaging a *block* of pixels. On the other hand, in dividing the H-transform by σ, we preserve the information on any object that is detectable by summing a block of pixels! The quantized H-transform preserves the mean of the image for every block of pixels having a mean significantly different than that of neighboring blocks of pixels.

If the noise is not constant across the image then this quantization method must be modified. The best approach we have found is to first scale the data to force the noise to be approximately constant in each pixel, and then to apply the H-transform and quantization described above. For CCD data, for example, the noise is a combination of Poisson counting statistics and readout noise. If we replace the input image I_{ij} by a scaled image $U_{ij} = 2\sqrt{I_{ij} + N^2}$, where N is the readout noise in each pixel, then the image U_{ij} has noise $\sigma_U \simeq 1$ in each pixel. U can then be compressed efficiently using the method described in this paper. Unfortunately, the use of this method for lossless compression is rather messy because considerable effort is required to make the square root transformation exactly reversible.

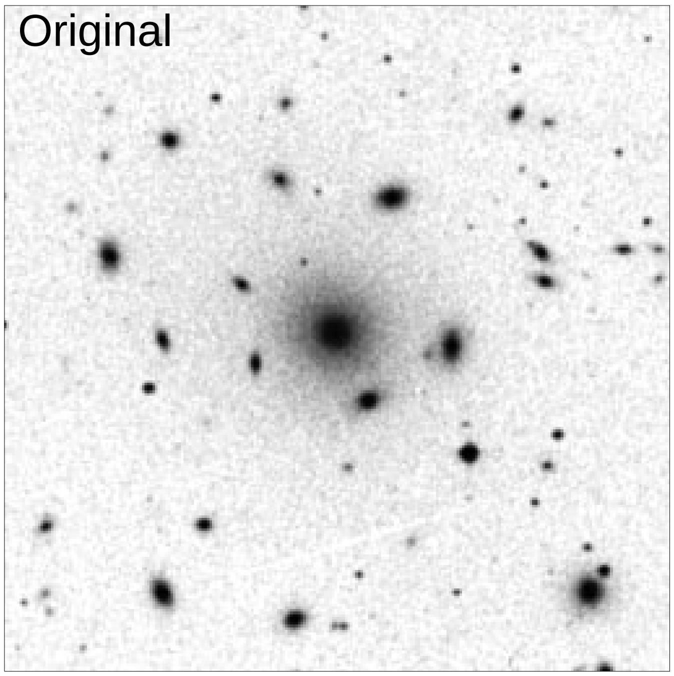

Figure 1. 256×256 pixel section of image from digitized Palomar Sky Survey E-plate of Coma galaxy cluster. Image has 16-bits; a logarithmic gray scale is used to make the noise near the sky brightness level visible.

4. QUADTREE CODING

The quantized H-transform has a rather peculiar structure. Not only are large areas of the transform image zero, but the non-zero values are strongly concentrated in the lower-order coefficients. The best approach we have found to code the coefficient values efficiently is quadtree coding of each bit-plane of the transform array. Quadtree coding has been used for many purposes[8]; the particular form we are using was suggested by Huang and Bijaoui[9] for image compression.

- Divide the bit-plane up into 4 quadrants. For each quadrant code a '1' if there are any 1-bits in the quadrant, else code a '0'.
- Subdivide each quadrant that is not all zero into 4 more pieces and code them similarly. Continue until one is down to the level of individual pixels.

This coding (which Huang and Bijaoui call "hierarchic 4-bit one" coding) is obviously very well suited to the H-transform image because successively lower orders of the H-transform coefficients are located in successively divided quadrants of the image.

We follow the quadtree coding with a fixed Huffman coding that uses 3 bits for quadtree values that are common (*e.g.*, 0001, 0010, 0100, and 1000) and uses 4 or 5 bits for less common values. This reduces the final compressed file size by about 10% at little computational cost. Slightly better compression can be achieved by following quadtree coding with arithmetic coding[10], but the CPU costs of arithmetic coding are not, in our application, justified for 3–4% better compression. We have also tried using arithmetic coding directly on the H-transform, with various contexts of neighboring pixels, but find it to be both computationally inefficient and not significantly better than quadtree coding.

For completely random bit-planes, quadtree coding can actually use more storage than simply writing the bit-plane directly; in that case we just dump the bit-plane with no coding.

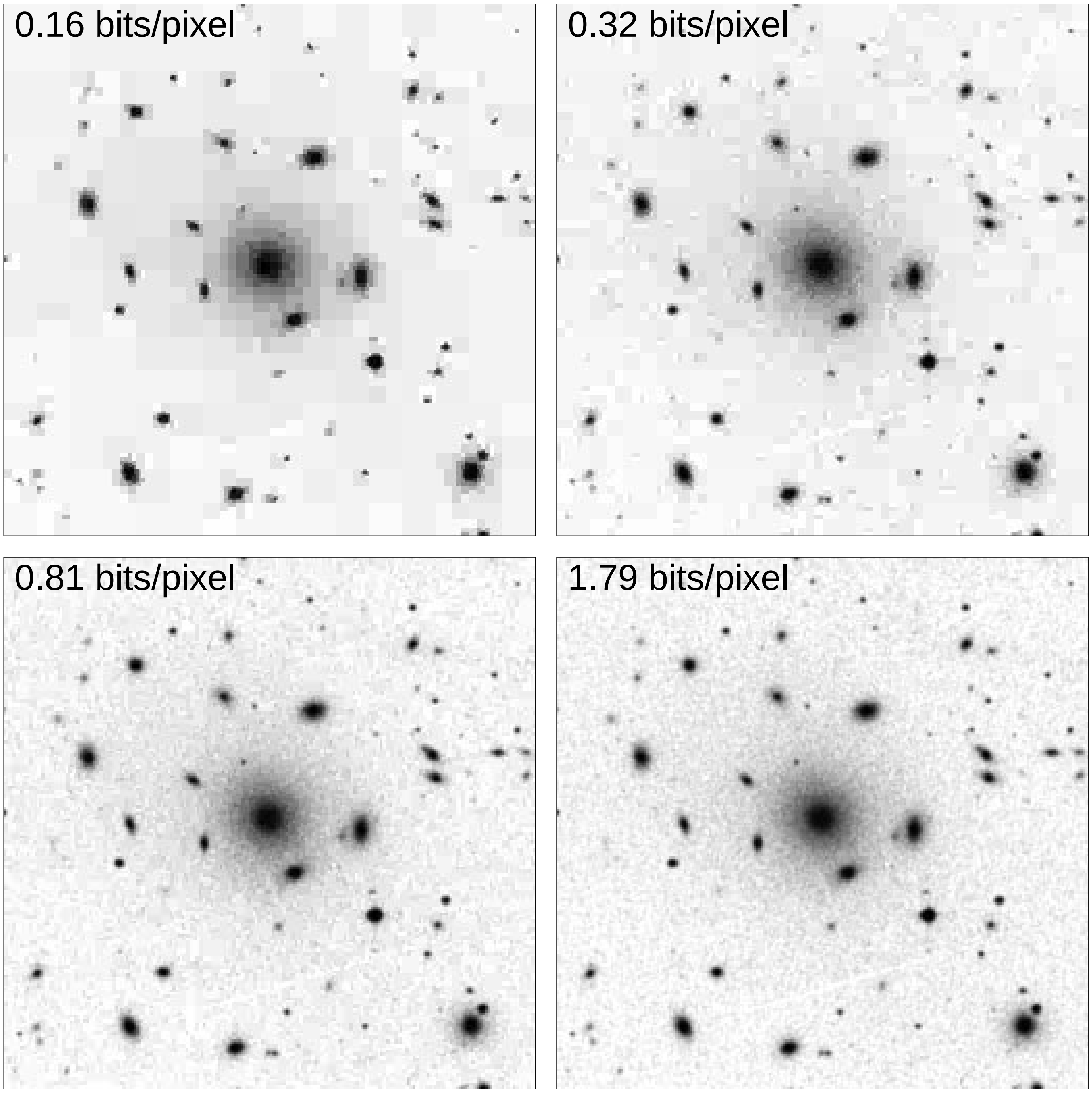

Figure 2. Effect of compression by H-transform and quadtree coding scheme described in this paper. This is also a sequence of images using the progressive transmission scheme; each image is the result of coding and transmitting another bit-plane from the H-transform.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.