10^{th} IMEKO TC7 International Symposium
June 30–July 2, 2004, Saint-Petersburg, Russia

HOW TO BEST COMPRESS 3-D MEASUREMENT DATA UNDER GIVEN GUARANTEED ACCURACY

Olga Kosheleva[1], *Sergio Cabrera*[1], *Brian Usevitch*[1], *Edward Vidal, Jr.*[2]

[1]Electrical & Computer Engr., University of Texas, El Paso, USA
[2]Army Research Laboratory, While Sands Missile Range, New Mexico, USA

Abstract. The existing image and data compression techniques try to minimize the mean square deviation between the original data $f(x,y,z)$ and the compressed-decompressed data $\widetilde{f}(x,y,z)$. In many practical situations, reconstruction that only guaranteed mean square error over the data set is unacceptable: for example, if we use the meteorological data to plan a best trajectory for a plane, what we really want to know are the meteorological parameters such as wind, temperature, and pressure along the trajectory. If along this line, the values are not reconstructed accurately enough, the plane may crash – and the fact that on average, we get a good reconstruction, does not help. What we need is a compression that guarantees that for each (x,y), the difference $|f(x,y,z)-\widetilde{f}(x,y,z)|$ is bounded by a given value Δ – i.e., that the actual value $f(x,y,z)$ belongs to the interval $[\widetilde{f}(x,y,z)-\Delta,\widetilde{f}(x,y,z)+\Delta]$. In this paper, we describe new efficient techniques for data compression under such interval uncertainty.

Keywords: data compression, guaranteed error bounds, meteorological data

1. FORMULATION OF THE PROBLEM

1.1. Compression is important

At present, so much data is coming from measuring instruments that it is necessary to compress this data before storing and processing. We can gain some storage space by using lossless compression, but often, this gain is not sufficient, so we must use lossy compression as well.

1.2. Successes of 2-D image compression

In the last decades, there has been a great progress in image and data compression. In particular, the JPEG2000 standard (see, e.g., [6]) uses the wavelet transform methods together with other efficient compression techniques to provide a very efficient compression of 2D images.

Within this standard, we can select different bitrates (i.e., number of bits per pixel that is required, on average, for the compressed image), and depending on the bitrate, get different degrees of compression.

When we select the highest possible bitrate, we get the lossless compressions that enables us to reconstruct the original image precisely. When we decrease the bitrate, we get a lossy compression; the smaller the bitrate, the more the compressed/decompressed image will differ from the original image.

1.3. 2-D data compression

In principle, it is possible to use these compression techniques to compress 2D measurement data as well.

1.4. Compressing 3-D data: layer-by-layer approach

It is also possible to compress 3D measurement data $f(x,y,z)$ – e.g., meteorological measurements taken in different places (x,y) at different heights z.

One possibility is simply to apply the 2D JPEG2000 compression to each horizontal layer $f(x,y,z_0)$.

1.5. Compressing 3-D data: an approach that uses KLT transform

Another possibility, in accordance with Part 2 of JPEG2000 standard, is to first apply the KLT transform to each vertical line. Specifically, we:

- compute the average value
$$\bar{f}(z)=\frac{1}{N}\cdot\sum_{x,y} f(x,y,z)$$
of the analyzed quantity at a given height z, where N is the overall number of horizontal points (x,y);
- compute the covariances $V(z_1,z_2)$ between different heights as
$$\frac{1}{N}\cdot\sum_{x,y}(f(x,y,z_1)-\bar{f}(z_1))\cdot(f(x,y,z_2)-\bar{f}(z_2));$$
- find the eigenvector λ_k and the eigenvectors $e_k(z)$ of the covariance matrix $V(z_1,z_2)$; we sort these eigenvalues into a sequence $e_1(z),e_2(z),\ldots$ so that $|\lambda_1|\ge|\lambda_2|\ge\ldots$;
- finally, we represent the original 3D data values $f(x,y,z)$ as a linear combination
$$f(x,y,z)=\bar{f}(z)+\sum_k a_k(x,y)\cdot e_k(z)$$
of the eigenvectors $e_k(z)$.

As a result, we represent the original 3-D data as a sequence of the horizontal *slices* $a_k(x,y)$:

- the first slice $a_1(x,y)$ corresponds to the main (1-st) eigenvalue;

- the second slice $a_2(x,y)$ corresponds to the next (2-nd) eigenvalue;
- etc.,

with the overall intensity decreasing from one slice to the next one.

Next, to each of these slices $a_k(x,y)$, we apply a 2D JPEG2000 compression with the appropriate bit rate b_k depending on the slice k.

1.6. Decompressing 3-D data: KLT-based approach
To reconstruct the data, we so the following:

- First, we apply JPEG2000 decompression to each slice; as a result, we get the values $\widetilde{a}_k^{[b_k]}(x,y)$.
- Second, based on these reconstructed slices, we now reconstruct the original 3-D data data as

$$\widetilde{f}(x,y,z)=\bar{f}(z)+\sum_k \widetilde{a}_k^{[b_k]}(x,y)\cdot e_k(z).$$

1.7. This approach is tailored towards image processing – and towards Mean Square Error
The problem with this approach is that for compressing measurement data, we use image compression techniques. The main objective of image compression is to retain the quality of the image. From the viewpoint of visual image quality, the image distortion can be reasonably well described by the mean square difference MSE (a.k.a. L^2-norm) between the original image $I(x,y)$ and the compressed-decompressed image $\widetilde{I}(x,y)$. As a result, sometimes, under the L^2-optimal compression, an image may be vastly distorted at some points (x,y) – and this is OK as long as the overall mean square error is small.

1.8. For data compression, MSE may be a bad criterion
When we compress measurement results, however, our objective is to be able to reproduce each individual measurement result with a certain guaranteed accuracy.

In such a case, reconstruction that only guaranteed mean square error over the data set is unacceptable: for example, if we use the meteorological data to plan a best trajectory for a plane, what we really want to know are the meteorological parameters such as wind, temperature, and pressure along the trajectory.

If along this line, the values are not reconstructed accurately enough, the plane may crash – and the fact that on average, we get a good reconstruction, does not help.

1.9. An appropriate criterion for data compression
What we need is a compression that guarantees the given accuracy for all pixels, i.e., that guarantees that the L^∞-norm $\max\limits_{x,y,z}|f(x,y,z)-\widetilde{f}(x,y,z)|$ is small.

1.10. What we need is data compression under interval uncertainty
In other words, what we need is a compression that guarantees that for each (x,y), the difference $|f(x,y,z)-\widetilde{f}(x,y,z)|$ is bounded by a given value Δ – i.e., that the actual value $f(x,y,z)$ belongs to the interval $[\widetilde{f}(x,y,z)-\Delta,\widetilde{f}(x,y,z)+\Delta]$.

There exist several compressions that provide such a guarantee. For example, if for each slice, we use the largest possible bitrate – corresponding to lossless compression – then $\widetilde{a}_k(x,y)=a_k(x,y)$ hence $\widetilde{f}(x,y,z)=f(x,y,z)$ – i.e., there is no distortion at all.

What we really want is, among all possible compression schemes that guarantee the given upper bound Δ on the compression/decompression error, to find the scheme for which the average bitrate

$$\bar{b}\overset{\text{def}}{=}\frac{1}{N_z}\cdot\sum_k b_k$$

is the smallest possible.

In some cases, the bandwidth is limited, i.e., we know the largest possible average bitrate b_0. In such cases, among all compression schemes with $\bar{b}\le b_0$, we must find a one for which the L^∞ compression/decompression error is the smallest possible.

1.11. What we have done
In this paper, we describe new efficient (suboptimal) techniques for data compression under such interval uncertainty.

2. NEW TECHNIQUE: MAIN IDEAS, DESCRIPTION, RESULTS

2.1. What exactly we do
Specifically, we have developed a new algorithm that uses JPEG2000 to compress 3D measurement data with guaranteed accuracy. We are following the general idea of Part 2 of JPEG2000 standard; our main contribution is designing an algorithm that selects bitrates leading to a minimization of L^∞ norm as opposed to the usual L^2-norm.

2.2. Let us start our analysis with a 2-D case
Before we describe how to compress 3-D data, let us consider a simpler case of compressing 2-D data $f(x,y)$. In this case, for each bitrate b, we can apply the JPEG2000 compression algorithm corresponding to this bitrate value. After compressing/decompressing the 2-D data, we get the values $\widetilde{f}^{[b]}(x,y)$ which are, in general, slightly different from the original values $f(x,y)$.

In the interval approach, we are interested in the L^∞ error

$$D(b)\overset{\text{def}}{=}\max_{x,y}\left|\widetilde{f}^{[b]}(x,y)-f(x,y)\right|.$$

The larger the bitrate b, the smaller the error $D(b)$. When the bitrate is high enough – so high that we can transmit all the data without any compression – the error $D(b)$ becomes 0.

Our objective is to find the smallest value b for which the L^∞ error $D(b)$ does not exceed the given threshold Δ. For the 2-D case, we can find this optimal $b_{\rm opt}$ by using the following iterative *bisection* algorithm. In the beginning, we know that the desired bitrate lies between 0 and the bitrate B corresponding to lossless compression; in other words, we know that $b \in [b^-, b^+]$, where $b^- = 0$ and $b^+ = B$.

On each iteration of the bisection algorithm, we start with an interval $[b^-, b^+]$ that contains $b_{\rm opt}$ and produce a new half-size interval still contains $b_{\rm opt}$. Specifically, we take a midpoint $b_{\rm mid} \stackrel{\rm def}{=} (b^- + b^+)/2$, apply the JPEG2000 compression with this bitrate, and estimate the corresponding value $D(b_{\rm mid})$. Then:

- If $D(b_{\rm mid}) \le \Delta$, this means that $b_{\rm opt} \le b_{\rm mid}$, so we can replace the original interval $[b^-, b^+]$ with the half-size interval $[b^-, b_{\rm mid}]$.
- If $D(b_{\rm mid}) > \Delta$, this means that $b_{\rm opt} > b_{\rm mid}$, so we can replace the original interval $[b^-, b^+]$ with the half-size interval $[b_{\rm mid}, b^+]$.

After each iteration, the size of the interval halves. Thus, after K iterations, we can determine $b_{\rm opt}$ with accuracy 2^{-K}.

2.3. 3-D problem is difficult

In the 3-D case, we want to find the bitrate allocation $b_1, \ldots, b_{N_z}$ that lead to the smallest average bit rate b among all the allocations that fit within the given interval, i.e., for which the L^∞ compression/decompression error does not exceed the given value Δ: $D(b_1, b_2, \ldots) \le \Delta$.

For each bitrate allocation, we can explicitly compute this error, but there are no analytic formulas that describe this dependence, so we end up having to optimize a complex function with a large number of variables b_i.

Such an optimization is known to be a very difficult task, because the computational complexity of most existing optimization algorithms grows exponentially with the number of variables. There are theoretical results showing that in general, this growth may be inevitable; to be more precise, this problem is known to be NP-hard; see, e.g., [8].

2.4. Our main idea: using the upper estimate (enclosure) for the optimized error function

In our case, the problem is, e.g., to find, among all bitrate allocations $(b_1, b_2, \ldots)$ with $\overline{b} \le b_0$, the one for which the L^∞ compression/decompression error $D(b_1, b_2, \ldots)$ is the smallest possible.

Since it is difficult to minimize the original function $D(b_1, \ldots)$, we find easier-to-optimize upper estimate $\widetilde{D}(b_1, b_2, \ldots) \ge D(b_1, b_2, \ldots)$ and then find the values b_i that minimize $\widetilde{D}(b_1, \ldots)$. As a result, we find an allocation b_i guaranteeing that $\widetilde{D}(b_1, \ldots) \le \widetilde{D}_{\min}$ and thus, that $D(b_1, \ldots) \le \widetilde{D}_{\min}$.

Since, in general, $D(b_1, \ldots) \le \widetilde{D}(b_1, \ldots)$, the resulting allocation is only *suboptimal* with respect to $D(b_1, \ldots)$.

2.5. Explicit formula for the enclosure

Since we use the KLT, the difference

$$f(x,y,z) - \widetilde{f}(x,y,z)$$

is equal to

$$\sum_k \left(a_k(x,y) - \widetilde{a}_k^{[b_k]}(x,y)\right) \cdot e_k(z).$$

Therefore, once we know the L^∞-norms

$$D_k(b_k) \stackrel{\rm def}{=} \max_{x,y} \left| a_k(x,y) - \widetilde{a}_k^{[b_k]}(x,y) \right|$$

of the compression/decompression errors of each slice, we can conclude that

$$\left| a_k(x,y) - \widetilde{a}_k^{[b_k]}(x,y) \right| \le D_k(b_k),$$

hence, that

$$\left| (a_k(x,y) - \widetilde{a}_k^{[b_k]}(x,y)) \cdot e_k(z) \right| \le D_k(b_k) \cdot E_k,$$

where $E_k \stackrel{\rm def}{=} \max\limits_z |e_k(z)|$. Thus, the desired L^∞ error is bounded by $\widetilde{D}(b_1, \ldots) \stackrel{\rm def}{=} \sum\limits_k D_k(b_k) \cdot E_k$,

2.6. Resulting algorithm: derivation

In accordance with the above idea, to get the (suboptimal) bitrate allocation b_i, we must minimize the function $\widetilde{D}(b_1, \ldots) = \sum\limits_k D_k(b_k) \cdot E_k$ under the condition that the $\sum\limits_k b_k = N_z \cdot b_0$. By using Lagrange multipliers, we can reduce this problem to the unconstrained optimization problem

$$\sum_k D_k(b_k) \cdot E_k + \lambda \cdot \sum_k b_k - N_z \cdot b_0 \to \min.$$

In the minimum, derivatives w.r.t. all the variables b_i should be 0s, so we end up with the equation $-D_k'(b_k) = \lambda / E_k$, where the Lagrange multiplier λ should be selected based on the value b_0.

It can be easily shown that the other problem – of minimizing the average bitrate under the constraint that the compression/decompression error does not exceed Δ – leads to the same equation.

As we have mentioned, the function $D_k(b)$ decreases when b increases, so $D_k'(b) < 0$, with $D_k'(b) \to 0$ as b grows. It is therefore reasonable to represent the desired equation as $|D_k'(b_k)| = \lambda / E_k$.

What are the bounds on λ? The larger b_k, the smaller λ.

From the above formula, we conclude that $\lambda = |D_k'(b_k)| \cdot E_k$, hence

$$\lambda \le \Lambda_k \stackrel{\rm def}{=} \left(\max_b |D_k'(b)| \right) \cdot E_k,$$

so $\lambda \leq \Lambda \stackrel{\text{def}}{=} \min\limits_{k} \Lambda_k$.

2.7. Algorithm: description

Once we know, for each slice k, the dependence $D_k(b)$ of the corresponding L^∞-error on the bitrate b, we can find the desired (suboptimal) values b_k as follows.

At first, we compute the above-described values Λ_k and Λ. We know that $\lambda \leq [\lambda^-, \lambda^+] := [0, \Lambda]$. We use bisection to sequentially halve the interval containing λ and eventually, find the optimal value λ.

Once we know an interval $[\lambda^-, \lambda^+]$ that contains λ, we pick its midpoint λ_{mid}, and then use bisection to find, for each k, the value b_k for which $|D'_k(b_k)| = \lambda_{\text{mid}}/E_k$. Based on these b_k, we compute the average bitrate $\overline{b}$. If $\overline{b} > b_0$, this means that we have chosen too small λ_{mid}, so we replace the original λ-interval with a half-size interval $[\lambda_{\text{mid}}, \lambda^+]$. Similarly, if $\overline{b} < b_0$, we replace the original λ-interval with a half-size interval $[\lambda^-, \lambda_{\text{mid}}]$.

After K iteration, we get λ with the accuracy 2^{-K}, so a few iterations are sufficient. Once we are done, the values b_k corresponding to the final λ_{mid} are returned as the desired bitrates.

The only remaining question is how to determine the dependence $D_k(b)$. In principle, we can try, for each slice k, all possible bitrates b, and thus get an empirical dependence $D_k(b)$.

We have shown, that this dependence can be described by the following analytical formula:

- $D_k(b) = A_1 \cdot (b - b_0)^\alpha$ for all the values b until a certain threshold t, and
- $D_k(b) = A_2 \cdot 2^{-b}$ for $b \geq t$.

This model is a slight modification of a model from [4]. So, instead of trying all possible values b, it is sufficient to try a few to determine the values of the parameters A_i, b_0, and α corresponding to the given slice. The threshold t can then be determined as the value for which $D_1(t) = D_2(t)$.

2.8. Results

To test our algorithm, we applied it to 3-D meteorological data: temperature T, pressure P, the components U, V, and W of the wind speed vector, and the water vapor ratio WV.

For meteorological data, the resulting compression indeed leads to a much smaller L^∞ error bound Δ_{new} than the L^∞ error bound Δ_{MSE} corresponding to the bitrate allocation that optimizes the MSE error. The ratio $\Delta_{\text{new}}/\Delta_{\text{MSE}}$ decreases from 0.7 for $b_0 = 0.1$ to 0.5 for $b_0 = 0.5$ to ≤ 0.1 for $b_0 \geq 1$.

3. OPTIMAL LAYER-BY-LAYER COMPRESSION

3.1. Why layer-by-layer compression?

In the previous sections, we first applied the KLT transform to decrease the size of the data, and then applied the JPEG2000 techniques to compress the resulting slices. In many practical situations, our only objective is to achieve as large a compression as possible. In such situations, it makes sense to apply the KLT transform, because this transform leads to an additional data compression.

In some practical situations, however, we may want not only to decompress the data, but also to apply some geometric transformations to the data: shift, rotate, and/or scale the data, select a region, keep the same accurate data at a certain region of interest (ROI) while ignoring the details outside our ROI, etc.

One of the major advantages of JPEG2000 compression is that this compression enables us to perform these transformations directly on the compressed data, without the need to first decompress the data; see, e.g., [5–7]. However, if we first use a KLT transform, we lose this ability. So, if we first apply KLT, then the only way to select a ROI (or apply any other geometric transformation) is to:

- first, decompress the data;
- then apply the desired transformation to the decompressed data, and
- finally, compress the data again.

In other words, in situations where the above-described geometric transformations may be necessary, if we use KLT transform as the first stage of 3-D data compression, we save storage space, but we drastically increase the processing time.

In many computer systems, storage space is not as important as it used to be, while processing time is very critical. In such systems, it is therefore preferable to avoid the time-increasing KLT transform and, instead, apply the JPEG200 compression to each 2-D layer.

In other words, we represent the 3-D data $f(x, y, z)$ as a collection of 2-D layers $f_z(x, y) \stackrel{\text{def}}{=} f(x, y, z)$ corresponding to different heights z. To each layer z, we apply JPEG2000 compression with an appropriate bitrate b_z.

3.2. Decompressing 3-D data: layer-by-layer approach

To reconstruct the data, we so the following:

- First, we apply JPEG2000 decompression to each layer; as a result, we get the values $\widetilde{f}_z^{[b_z]}(x, y)$.
- Second, based on these reconstructed layers, we now reconstruct the original 3-D data data as $\widetilde{f}^{[b]}(x, y, z) = \widetilde{f}_z^{[b_z]}(x, y)$, where $b \stackrel{\text{def}}{=} (b_1, b_2, \ldots)$ is the corresponding bitrate allocation.

3.3. Formulation of the corresponding optimization problems

Similar to the KLT case, it is reasonable to consider two related optimization problems.

In the first problem, among all possible compression schemes that guarantee the given upper bound Δ

on the compression/decompression error, i.e., for which $D(b) \le \Delta$, where

$$D(b) \stackrel{\rm def}{=} \max_{x,y,z} \left| f(x,y,z) - \widetilde{f}^{[b]}(x,y,z) \right|,$$

we want to find the bitrate allocation $b = (b_1, b_2, \ldots)$ for which the average bitrate

$$\overline{b} \stackrel{\rm def}{=} \frac{1}{N_z} \cdot \sum_z b_z$$

is the smallest possible.

In some cases, we know the largest possible average bitrate b_0. In such cases, we have another optimization problem: among all compression schemes b with $\overline{b} \le b_0$, we must find a one for which the compression/decompression error $D(b)$ is the smallest possible.

3.4. Towards the solution of the optimization problem: reducing the overall error to layer-by-layer errors

As we mentioned in Section 2, we can determine how the compression-decompression error

$$D_z(b_z) \stackrel{\rm def}{=} \max_{x,y} \left| f_z(x,y) - \widetilde{f}_z^{[b_z]}(x,y) \right|$$

of each layer z depends on the corresponding bitrate b_z.

By definition, $D(b)$ is the largest of all the differences $\left| f(x,y,z) - \widetilde{f}^{[b]}(x,y,z) \right|$ over all all points (x,y,z) in 3-D space. If we fix z and take maximum over all x and y, then we get $D_z(b_z)$. Thus, $D(b) = \max\limits_z D_z(b_z)$.

3.4. The corresponding optimization problems are non-smooth, hence difficult to solve

In Section 2, we used the Lagrange multiplier method to solve the corresponding optimization problem. The possibility to use this method was based on the fact that the dependence of the optimized function $\widetilde{D}(b_1,\ldots) = \sum\limits_k D_k(b_k) \cdot E_k$ on bitrates b_z is differentiable.

In our case, the expression $D(b) = \max\limits_z D_z(b_z)$ contains the function $\max$ which is, in general, not differentiable. As a result, we cannot use the above techniques to optimize $D(b)$.

3.5. New idea: description

Instead, we propose (and justify) the following simple idea. Let us first describe this idea on the example of the first optimization problem, i.e., the problem of minimizing the average bit rate $\overline{b}$ under the constraint $D(b) = \max\limits_z D_z(b_z) \le \Delta$.

It turns out that the solution to this problem can be obtained by solving, for every z, the equation $D_z(b_z) = \Delta$.

In the second optimization problem, we are given the maximum allowed average bitrate b_0, and we must find, among all bitrate allocations b for which $\overline{b} \le b_0$, an allocation with the smallest value $D(b)$.

We claim that the solution to this problem can be obtained, for some value Δ, by solving, for every z, the equation $D_z(b_z) = \Delta$.

Let us prove these two claims.

3.6. New idea: justification for the first optimization problem

Let us first prove the statement about the first optimization problem. Indeed, we are given the value Δ such that $D(b) = \max\limits_z D_z(b_z)$ cannot exceed Δ. Since the largest of the values $D_z(b_z)$ cannot exceed Δ, this means that for each layer z, we must select a bit rate b_z for which the corresponding value $D_z(b_z)$ does not exceed Δ.

Our objective is to minimize the average bit rate. Overall, the smaller the bit rate b_z, the larger the distortion, i.e., the larger the compression/decompression error $D_z(b_z)$. Thus, to attain the smallest possible value of the bit rate b_z, we must use the largest possible value of $D_z(b_z)$ – i.e., we must select b_z in such a way that $D_z(b_z) = \Delta$. The statement is proven.

3.7. New idea: justification for the second optimization problem

Let us prove the claim about the second optimization problem. Indeed, for every value of Δ, we can find b_z for which $D_z(b_z) = \Delta$. Let us select Δ in such a way that for the corresponding b_z, the average bit rate is exactly b_0. Let $\widetilde{b}_z$ denote the values b_z corresponding to thus selected Δ, i.e., the values for which $D_z(\widetilde{b}_z) = \Delta$. Let us prove, by reduction to a contradiction, that these values $\widetilde{b}_z$ are indeed optimal. Indeed, let us assume that they are not optimal. This means that there exist different values of bit rates $b'_1, \ldots, b'_{N_z}$ for which the average bit rate is the same, i.e., $\sum\limits_{z=1}^{N_z} b'_z = \sum\limits_{z=1}^{N_z} \widetilde{b}_z$, but for which

$$\max_z D_z(b'_z) < \max_z D_z(\widetilde{b}_z). \qquad (1)$$

We know that some of the bit rates b'_z are different from the corresponding bit rates $\widetilde{b}_z$. The values b'_z cannot be all larger or equal than the corresponding values $\widetilde{b}_z$ because then the averages would not be equal. Therefore, there is at least one layer z_0 for which $b'_{z_0} < \widetilde{b}_{z_0}$. Since the function $D_z(b)$ is decreasing, we conclude that $D_{z_0}(b'_{z_0}) > D_{z_0}(\widetilde{b}_{z_0})$. We have selected $\widetilde{b}_z$ in such a way that $D_{z_0}(\widetilde{b}_{z_0}) = \Delta = \max\limits_z D_z(\widetilde{b}_z)$. Thus, we can conclude that $D_{z_0}(b'_{z_0}) > \max\limits_z D_z(\widetilde{b}_z)$, and therefore, $\max\limits_z D_z(b'_z) \ge D_{z_0}(b'_{z_0}) > \max\limits_z D_z(\widetilde{b}_z)$, which contradicts to our assumption (1). This contradiction shows that the values b_z for which $D_z(b_z) = \Delta$ are indeed optimal. The statement is proven.

3.8. Resulting algorithm: first optimization problem

For the first optimization problem, we must find, for each z, the value b_z for which $D_z(b_z) = \Delta$.

To get the most accurate description of how distortions $D_z(b)$ depend on the bitrate b, we can experimentally determine, for each layer z, the corresponding rate distortion curve $D_z(b)$. Once we know these curves, we determine, for each z, the value b_z by using bisection (as in Section 2.2).

Alternatively, we can use the analytical expressions similar to the ones described in Section 2.7. In that section, we describe the expression for the slices resulting from the KLT transform; for non-KLT layers, however, we get a similar expression. In this case, we can have an explicit expression for b_z:

- if $\Delta \le A_2 \cdot 2^{-t}$, then
$$b_z(\Delta) = -\log_2\left(\frac{\Delta}{A_2}\right);$$
- if $\Delta \ge A_2 \cdot 2^{-t}$, then
$$b_z(\Delta) = b_0 + \left(\frac{\Delta}{A_1}\right)^{1/\alpha}.$$

3.9. Resulting algorithm: second optimization problem

For each Δ, we can apply the algorithm from the previous section to find the corresponding values $b_z(\Delta)$, and then compute the corresponding average bitrate
$$\bar{b}(\Delta) = \frac{1}{N_z}\sum_{z=1}^{N_z} b_z(\Delta).$$
Our objective is, given b_0, to find the value Δ_{opt} for which $\bar{b}(\Delta_{\text{opt}}) = b_0$. We will find this value Δ by using bisection.

The smaller the distortion Δ, the larger the bitrates b_z that are needed to achieve this distortion level, so the larger the average bitrate $\bar{b}$.

We start with two values of Δ: a very small value Δ^- for which $\bar{b}(\Delta^-) > b_0$, and a very large value Δ^+ for which $\bar{b}(\Delta^+) < b_0$. We know that the desired value Δ_{opt} lies in the interval $[\Delta^-, \Delta^+]$.

On each iteration of the bisection algorithm, we start with an interval $[\Delta^-, \Delta^+]$ that contains Δ_{opt} and produce a new half-size interval still contains Δ_{opt}. Specifically, we take a midpoint
$$\Delta_{\text{mid}} \overset{\text{def}}{=} \frac{\Delta^- + \Delta^+}{2},$$
and apply the above algorithm to compute the corresponding value $\bar{b}(\Delta_{\text{mid}})$. Then:

- If $\bar{b}(\Delta_{\text{mid}}) \le b_0$, this means that $\Delta_{\text{opt}} \le \Delta_{\text{mid}}$, so we can replace the original interval $[\Delta^-, \Delta^+]$ with the half-size interval $[\Delta^-, \Delta_{\text{mid}}]$.
- If $\bar{b}(\Delta_{\text{mid}}) > b_0$, this means that $\Delta_{\text{opt}} > \Delta_{\text{mid}}$, so we can replace the original interval $[\Delta^-, \Delta^+]$ with the half-size interval $[\Delta_{\text{mid}}, \Delta^+]$.

After each iteration, the size of the interval halves. Thus, after K iterations, we can determine Δ_{opt} with accuracy 2^{-K}.

3.10. Results

To test our algorithms, we applied them to the 3-D meteorological data described in Section 2.8. Similarly to Section 2, for this meteorological data, the resulting compression leads to a much smaller L^∞ error bound Δ_{new} than the L^∞ error bound Δ_{MSE} corresponding to the bitrate allocation that optimizes the MSE error.

ACKNOWLEDGMENTS

This work was partially supported by NSF Grant CDA-9522207, the ARO grants DAA H04-95-1-0494 and DAA D19-99-1-0012, by the Texas Instruments Grant, and by NASA under cooperative agreement NCC5-209.

This work was partly performed during O.K.'s visit to Brazil; this visit was sponsored by the Brazilian funding agency CTINFO/CNPq.

REFERENCES

[1] S.D. Cabrera, "Three-Dimensional Compression of Mesoscale Meteorological Data based on JPEG2000", *Battlespace Digitization and Network-Centric Warfare II*, Proc. SPIE, **4741**, 239–250, 2002.

[2] O.M. Kosheleva, *Task-Specific Metrics and Optimized Rate Allocation Applied to Part 2 of JPEG2000 and 3-D Meteorological Data*, Ph.D. Dissertation, University of Texas at El Paso, 2003.

[3] O.M. Kosheleva, B. Usevitch, S. Cabrera, E. Vidal, Jr., "MSE Optimal Bit Allocation in the Application of JPEG2000 Part 2 to Meteorological Data", *Proceedings of the 2004 IEEE Data Compression Conference DCC'2004*, Snowbird, Utah, March 23–25, 2004, p. 546.

[4] S. Mallat, F. Falzon, "Analysis of low bit rate image transform coding", *IEEE Trans. Signal Proc.*, **46**, 1027–1042, 1998.

[5] D.S. Taubman, R. Leung, A. Sekecr, "Scalable compression of volumetric images", *Proc. ICIP*, SEptember 2002, pp. 22–25.

[6] D.S. Taubman, M.W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*, Kluwer, Boston, Dordrecht, London, 2002.

[7] D.S. Taubman, M.W. Marcellin, "JPEG2000: standard for interactive imaging", *Proc. IEEE*, **90**(8), 1336–1357, 2002.

[8] S.A. Vavasis, *Nonlinear Optimization: Complexity Issues*, Oxford University Press, New York, 1991.

Authors: Olga Kosheleva, Sergio Cabrera, Brian Usevitch, Department of Electrical and Computer Engineering, University of Texas at El Paso, 500 W. University, El Paso, TX 79968, USA, corresponding author O.K. phone 1-915-747-7588, fax 1-915-747-5030, email olgak@utep.edu.

Edward Vidal, Jr., Army Research Laboratory, While Sands Missile Range, NM 88002-5501, USA, email evidal@arl.army.mil.