Stanford Exploration Project, Report 125, January 16, 2007, pages 1–??

Covariance based interpolation

Madhav Vyas

ABSTRACT

I test and extend an image interpolation algorithm designed for digital images to suite seismic data sets. The problem of data interpolation can be addressed by determining a filter based on global and local covariance estimates. Covariance estimates have enough information to discern the presence of sharp discontinuities (edges) without the need to explicitly determine the dips. The proposed approach has given encouraging results for a variety of textures and seismic data sets. However, when sampling is too coarse (aliasing) a proxy data set needs to be introduced as an intermediate step. In images with bad signal-to-noise ratio, covariance captures the trend of the signal as well as that of the noise; to handle such situations, a model-styling goal (regularization) is incorporated within the interpolation scheme. Various test cases are illustrated in this article, including one using post-stack 3D data from the Gulf of Mexico.

INTRODUCTION

The problem of interpolation arises because the geometry and spacing of the data acquisition grid differs from that required ideally for seismic imaging. In most cases, either the data has some holes (missing data), or sampling is too coarse for high-resolution seismic imaging. By the very nature of the problem we can say that no interpolation scheme can be perfect in all respects, because a proxy can never substitute for the real data. The criterion normally used to judge the performance of any interpolation scheme is how well it resembles the real data. Hence most interpolation schemes exploit the information present in the spectrum of the data and make the inserted values conform to the same spectrum. Prediction Error Filters (PEF) exactly rely on this principle (Claerbout, 2005). Covariance in the data space is analogous to the spectrum, and in the method presented in this article I use this covariance structure to estimate the filter coefficients and subsequently fill in the missing values.

A general problem with various interpolation schemes is that they do not perform well in presence of sharp features (high-frequency components), and the image after interpolation gets smeared. A common solution is to estimate edges and dips explicitly and use this information while interpolating; this ensures good resolution along the estimated features. Two important drawbacks of this approach are first, that we are restricted to a finite number of choices, and second, that it leads to increased computational complexity. Li and Orchard (2001) proposed a covariance-based interpolation scheme especially for digital images. Their claim is that covariance has enough information that an interpolation scheme based on covariance structure

can preserve the edges and sharp features without a need to explicitly estimate them.

In this article I explore the scope of application for such an algorithm in the context of seismic data, which must be handled somewhat differently from digital images. I extend the algorithm to incorporate the idea of intermediate proxy data (to handle aliasing) and model-styling. Interpolation of 3D seismic data is challenging because of the increased computational cost and complexity in estimating covariance matrices and filter coefficients. I also extend the method to work for 3D data sets, results obtained for post-stack 3D data from Gulf of Mexico are presented at the end.

THEORY

The problem of data interpolation can be thought of as re-sampling an image or data set to a lower resolution (everything is known) and then transforming it back to a higher resolution image or data set. To formulate the problem, let us consider that we are trying to estimate a high-resolution image $Y(i,j)$ of dimension $2n \times 2n$ from a given low-resolution image $X(i,j)$ of dimension, $n \times n$. We assume that the even pixels in the high resolution image $(2i,2j)$ directly come from the pixels of the low-resolution image (i,j)(as given by equation 1). Shaded pixels in Figure 1 correspond to the known pixels and they directly come from the low resolution image.

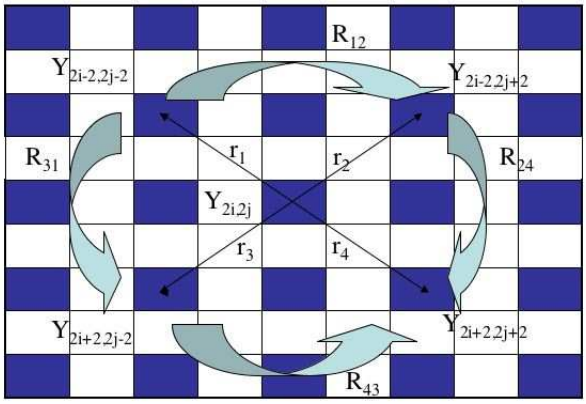

Figure 1: High resolution image Y, where shaded pixels represent the known values. R_{ij} and r_k are elements of matrices quantifying covariance amongst diagonal neighbors and between the central pixel and diagonal neighbors respectively. madhav1-figure1 [NR]

$$Y_{2i,2j} = X_{i,j} \tag{1}$$

Our goal is to determine the odd pixels $(2i+1,2j+1)$. Apart from these two classes of pixels where both the coordinates are either odd or even, we have one more class where one coordinate is odd and one is even $((2i+1,2j)$ or $(2i,2j+1))$. Estimating values of these pixels is similar to what is proposed below, but slightly different in terms of geometry of the filter. In

seismology, we normally do not have missing rows (time axis), and hence theory developed for determining odd pixels applies directly to seismic data sets.

The value of a missing pixel in image Y is estimated using its four diagonal neighbors and corresponding filter coefficients (α), as described by equation 2:

$$Y_{2i+1,2j+1} = \sum_{k=0}^{1}\sum_{l=0}^{1} \alpha_{2k+l} Y_{2(i+k),2(j+l)}. \tag{2}$$

Geometrically, the filter can be visualized as a convolution template, with 1 at the center and four filter coefficients on the corners. While interpolating, 1 sits on the position of the unknown pixel, and the filter coefficients on corresponding diagonal neighbors.

If we model these images as a locally stationary Gaussian process, according to Weiner filtering theory these interpolation coefficients are given by

$$\alpha = \mathbf{R}^{-1}\mathbf{r}, \tag{3}$$

where α is a vector containing filter coefficients, $\mathbf{R}$ is the local covariance matrix quantifying covariance between each of the diagonal neighbors, and $\mathbf{r}$ is the covariance vector characterizing covariance between the unknown pixel and four diagonal neighbors. Figure 1 displays elements of $\mathbf{R}$ and $\mathbf{r}$. We can calculate these matrices for a low-resolution image where all the pixels are known, but we can't directly compute these matrices for the high-resolution image that we desire, since many pixels are unknown. If we assume the statistics to be Gaussian in nature and the sampling distance to be reasonably small, we can approximate high-resolution covariance matrices by low-resolution covariance matrices (Li and Orchard, 2001). Claerbout (2005) discusses patching technology which can be used to break a section or data cube into multiple overlapping patches or windows, which can be considered stationary. If we consider a window of size M by M, it will have a total of M^2 pixels and each pixel will have four neighbors. Let $\mathbf{C}$ be a 4-by-M^2 matrix composed of diagonal neighbors of all the pixels, and $\mathbf{y}$ be a vector containing all these pixels; then matrices $\mathbf{R}$ and $\mathbf{r}$ of equation 3 are given by

$$\mathbf{R} = \frac{1}{M^2}\mathbf{C^T C}, \tag{4}$$

$$\mathbf{r} = \frac{1}{M^2}\mathbf{C^T y}. \tag{5}$$

Substituting these values we get,

$$\alpha = (\mathbf{C^T C})^{-1}(\mathbf{C^T y}). \tag{6}$$

In short, we first estimate covariance matrices on the low-resolution image and then solve for the filter coefficients. Once we know these filter coefficients, we can slide this filter throughout the image and estimate the missing values; the only condition is that all four legs of this filter should sit on known points. On some occasions we might need to scale the size

of the filter up or down (multi-scale) to touch the nearest non-zero diagonal neighbors; this is discussed at length in the section dealing with missing chunks of data.

Lets consider the case of alternating missing columns for the sake of demonstrating the method. The problem of alternate missing columns can be thought of as a problem of increasing the image resolution along one dimension. Let P be a seismic section of dimension $2n \times 2n$ with all the odd columns missing. This can be reduced to a section of size $2n \times n$ by compressing along the missing axis and removing the missing columns. We can now estimate the covariance values and filter coefficients (α) on this low-resolution section, and then use these estimates to calculate the pixels belonging to the missing columns as given below,

$$P_{i,2j+1} = \alpha_1 P_{i-2,2j} + \alpha_2 P_{i+2,2j} + \alpha_3 P_{i-2,2j+2} + \alpha_4 P_{i+2,2j+2}. \tag{7}$$

Notice that in equation 7 we are using pixels spaced one point apart from the missing point along columns and two points apart along rows. This was done to maintain the same directional covariance, since we distorted the image while compressing and throwing away zero values. We need to account for the asymmetry we introduced while estimating the filter coefficients. Whenever we have missing data points and we make a low-resolution image by compressing and throwing away those points, we should always account for the geometry.

2D TEST CASES

In this section I test the given algorithm on textures and on some real and synthetic seismic data sets. I address two different kinds of missing data problems, one where we have alternate missing traces and another where we have large chunks of missing data. Although, the method is not designed for missing chunks of data, as should be clear from the theory, with a few modifications and tricks we adapt the method for that use. The biggest problem for any interpolation scheme is to try to interpolate beyond aliasing; the presented method in its current form fails to accomplish this, but introduction of some intermediate proxy data before filter estimation has improved the results. In this section I discuss all these issues in detail with help of some examples.

Alternate Missing Columns

The problem of alternate missing columns (without aliasing) is one of the ideal cases to work on, especially, with this method. Interpolation using this approach involves two steps, first to estimate the filter using inversion, and second to estimate the missing data. The second step is not an inversion, unlike PEFs. Even though this has a computational advantage, it is not appropriate for real data sets, which might need some kind of model styling.

At first, I use the wood texture (Figure 2(a)) as a test case. In terms of statistics, this image is relatively stationary. I replace alternate columns with zero traces to create Figure 2(b) and then use the covariance-based filters to fill in the gaps. The reconstructed image is given in Figure 2(c). Results look good, primarily because we are very close to our approximation of

Figure 2: (a) Wood texture, (b) wood texture with alternate columns missing, and (c) reconstructed wood texture.

local stationarity. Another test case that I demonstrate here is the texture made up of ridges (Figure 3(a)) . This to some extent violates the assumption of stationarity because of the presence of multiple and conflicting dips. Figures 3(b) and 3(c) illustrate the images with missing and reconstructed data respectively. As can be observed, results are good but not perfect.

Missing Chunks of data

In this section we look at some cases where instead of alternate missing traces we have big chunks of missing data. In the approach described above, the interpolation filter makes use of the four nearest diagonal neighbors to estimate the missing values, but in cases of big chunks of missing data, this approach can not be extended directly, due to lack of any diagonal neighbor in the middle of the hole. What we can do alternatively is to use nearest non-zero diagonal neighbors instead of nearest diagonal neighbors, maintaining the symmetry so that we maintain the same directional covariances. Equation 8 describes how we can use columns situated far apart for interpolation.

$$P_{i,j} = \alpha_1 P_{i-h,j-h} + \alpha_2 P_{i+h,j-h} + \alpha_3 P_{i-h,j+h} + \alpha_4 P_{i+h,j+h}, \tag{8}$$

where h is the distance to nearest non-zero column. Figure 4(a) shows a texture made up of bricks, which again is more or less stationary, complying with the initial assumptions. If we create a hole of missing data as shown in Figure 4(b) and use a filter which instead of using nearest diagonal neighbors uses the tenth nearest diagonal neighbor (still a four point filter but ten times larger in size, as the coefficients are spaced ten points apart), we do a reasonable job of estimating the missing values as displayed in figure 5(a). Since the filter

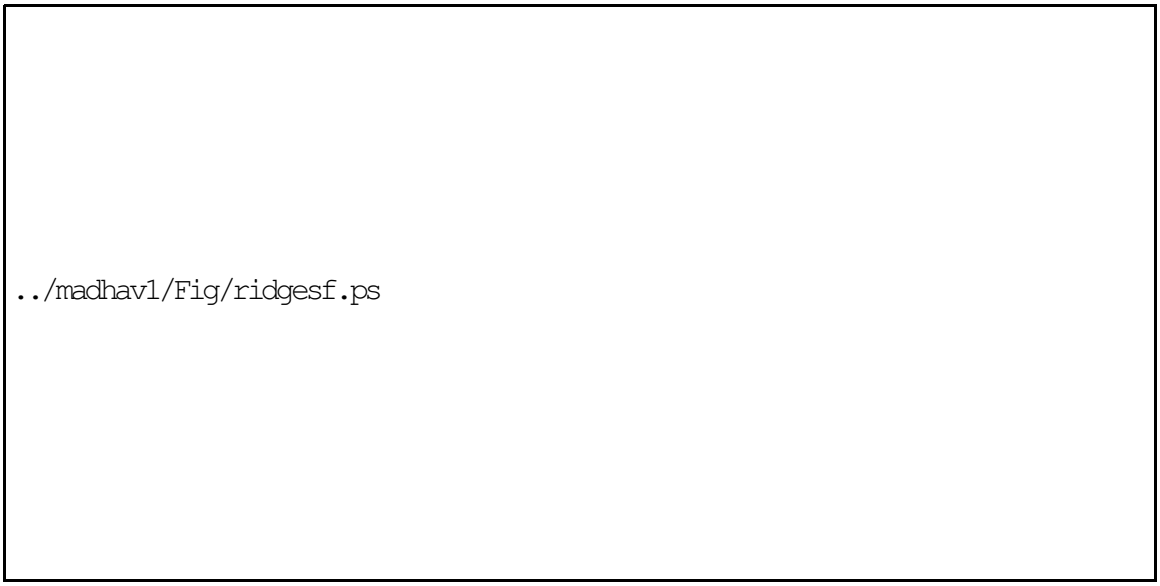

Figure 3: (a) Ridge texture, (b) ridge texture with alternate columns missing, and (c) reconstructed ridges texture.

is much larger, it has pronounced boundary effects as can be noticed. Comparing the results obtained using this approach with those obtained with PEF (figure 5(b)), the main difference is that this approach maintains the same covariance within the missing chunk as measured from the data, whereas missing values obtained by PEF appear to be of lower frequency than actual values. 2D PEFs of size 5×5 with 500 iterations were used for all the examples presented in this section. Incidentally, the brick example happens to be a case where everything required by the covariance-based interpolation algorithm falls in place and results are good, it does not work equally well for all the cases in general, as discussed with the help of later examples.

This approach of multi-scaling worked well for the brick example in spite of scaling the filter by a factor of 10 (incidentally), but it has a severe limitation. When we use values situated far apart from actual point of interest, we by no means ensure continuity at the edges of the hole. This is illustrated with the help of the previously used wood texture. Figure 6 shows the missing data, and 7(a) is the image after interpolation. A sharp discontinuity is visible at the edges of the hole. In this case, the results of PEF are much better, as can be observed from Figure 7(b). PEF in this case performs better, primarily because it formulates the problem of filling the missing values as an inverse problem and introduces some kind of regularization. On the other hand in the covariance-based approach, estimation of missing values is just a one-step process, and no constraints can be levied. Only thing to appreciate about filling with the covariance-based approach is that it maintains the same texture in terms of frequency content.

There are different ways to attack the problem of bigger sized holes. One approach is to use an adaptable filter to interpolate, rather than a fixed large filter as was used above. In this case we do not ensure that filter is scaled equally in all directions; rather we let it expand independently in all directions until it hits a non-zero value, then freeze the size there and use

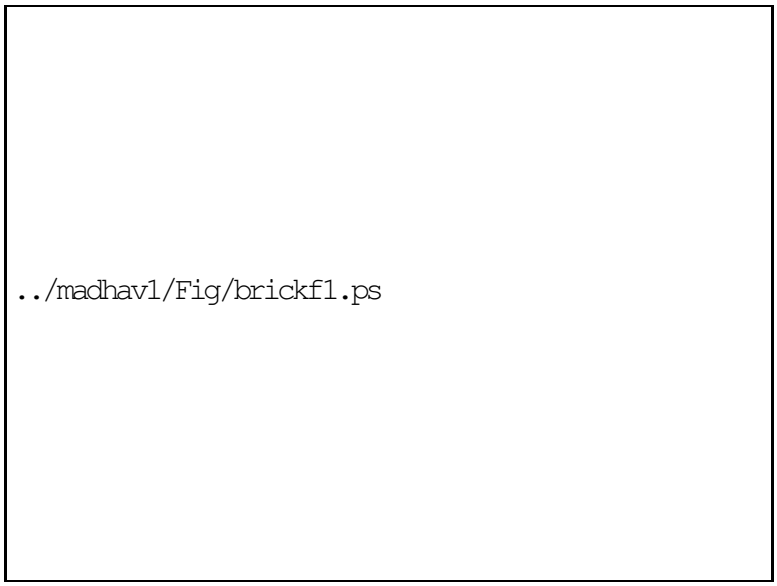

Figure 4: (a)Texture made up of bricks, and (b) texture with a missing block of data.

Figure 5: Reconstructed brick texture with (a) covariance-based approach, and (b) PEF.

Figure 6: Wood texture with missing hole of data.

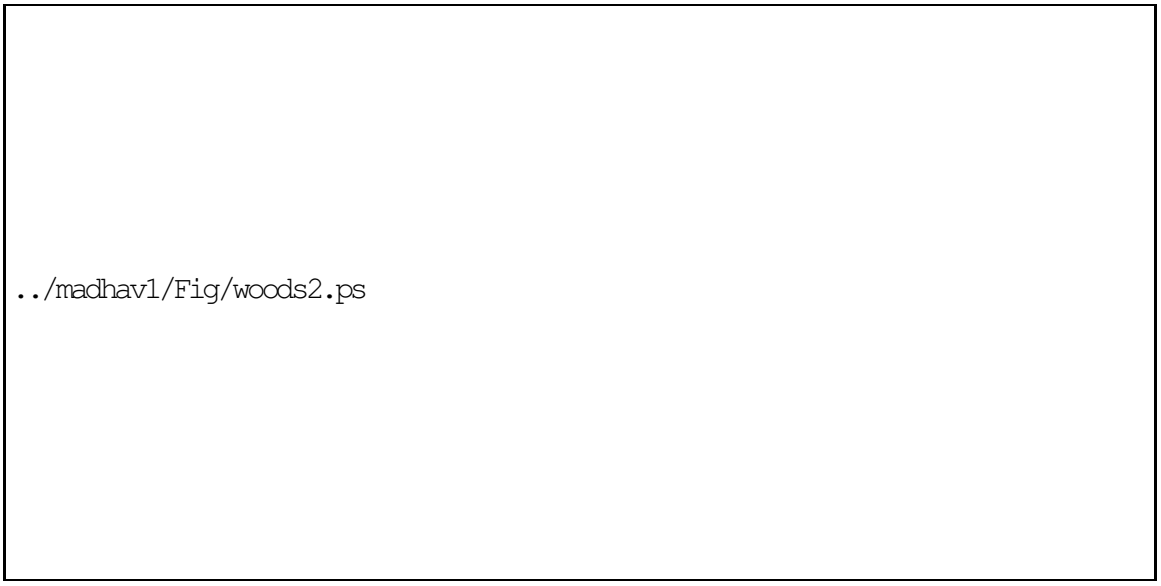

Figure 7: Reconstructed wood texture interpolated with (a) current approach, (b) PEF, and (c) modified iterative approach.

that filter to interpolate. Results obtained using this method were not encouraging, possibly because we do not maintain symmetry in the filter. Symmetry should be maintained at all scales to preserve the directional covariance.

Another approach is to interpolate iteratively. Consider a situation where the hole is of size h and is located at the jth position from left. When we interpolate any pixel belonging to the jth column using an unscaled filter and its four diagonal neighbors, we encounter two missing values corresponding to column $j+1$, and two known values corresponding to column $j-1$. As a result of this interpolation, we get nonzero values in column j and in column $j+h$ (which is at the other side of the hole). Although this value is a step in the right direction, it is far from the actual value. We can repeat this process until we completely fill the hole.

Results of the iterative scheme described above for the same wood texture are given in Figure 7(c). As can be seen, results are marginally better and maintain the image continuity, but they are still far from the true value. The results of the same experiment carried out on the ridges texture are given in Figure 8. Figure 8(a) shows the missing data, Figure 8(b) shows the result of interpolating with the iterative covariance-based scheme, and Figure 8(c) shows the result of interpolating with PEFs. On lower part of the hole iterative scheme proposed here seems to work a little better than PEFs in terms of getting the curvature and also, the amplitudes in middle of the hole look a little better in the results obtained by iterative covariance-based algorithm.

Figure 8: (a) Missing data in ridges, reconstructed with (b) modified iterative scheme and (c) PEF.

Interpolation beyond aliasing

The most challenging part of the interpolation problem is to go beyond aliasing. This proves to be the acid test for various interpolation schemes. The method proposed in this paper has

difficulties going beyond aliasing, as is illustrated with the help of a few examples. Figure 9(a) shows a synthetic data set (called "hask", a CMP gather with a hole) which I will use to test the method. Figure 9(b) shows the missing data set created from the original by throwing away every alternate trace, and the result of interpolation is given in Figure 9(c). As can be seen, results are poor, the covariance-based filter catches the aliased trends. Another example for which the method fails is that of three plane waves. Figure 10 shows the original data set, the data set with missing traces, and the interpolated result. In this case too, we end up estimating the covariance and filter based on the aliased trends. Stationary 2D PEFs (different shapes and sizes) also failed to interpolate satisfactorily for the synthetic data set, results for a 10×5 PEF with 1000 iterations are given in Figure 11(a). PEFs performed well on the example of three plane waves, as demonstrated in Figure 11(b).

The results presented above might suggest that the method cannot handle crossing or multiple dips. This would definitely be unfortunate, especially in the context of seismic data which need not have regular, pre-defined patterns. I show with the help of a few examples that it is the aliasing and not the presence of multiple dips that is the source of problem. Figure 12(a) is the same synthetic data set that was used above, with zero traces inserted in between; as a result the data is not aliased but has multiple and conflicting dips. Figure 12(b) is the interpolated result. Results of a similar exercise for the plane-wave example are given in Figure 13.

Figure 9: (a) Synthetic data set (called "hask"), (b) alternate missing traces, and (c) result after interpolation.

Aliasing is related to both the steepness of dips and the sampling rate. One way to get around this problem, at least with CMP gathers, is to apply interpolation after flattening using NMO (Ji and Claerbout, 1991). In an ideal case when velocity information is perfect, gathers are absolutely flat, but even if they are not, anything approaching flat gathers is good enough for interpolation. In the limiting case where everything is perfectly flat, linear interpolation

Figure 10: (a) Three plane waves, (b) alternate traces missing, and (c) result after interpolation.

Figure 11: Results obtained with PEF for (a) hask and (b) plane waves.

Figure 12: (a) Synthetic data set with zero traces inserted in between, and (b) reconstructed result.

Figure 13: (a) Three plane waves with zero traces inserted in between, and (b) reconstructed result.

will do as well as any other method. Figure 14(a) shows NMOed missing data corresponding to Figure 9(b), and figure 14(b) shows the result of interpolation.

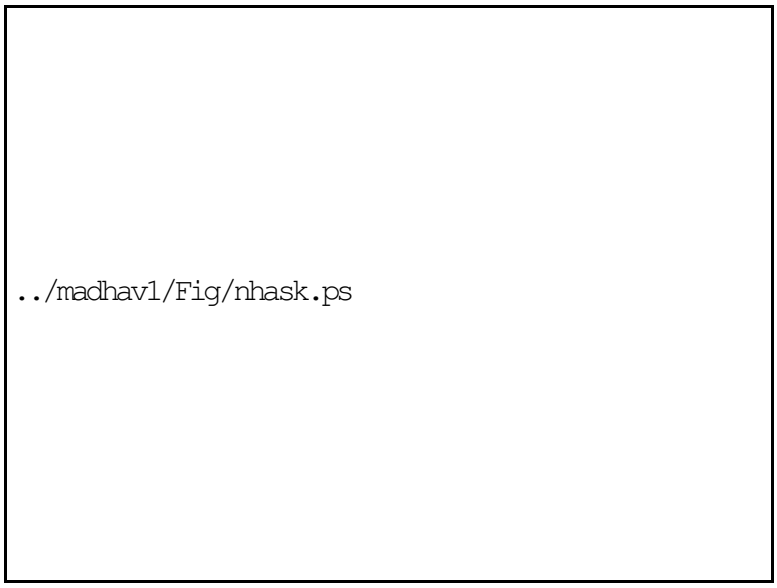

Figure 14: (a) NMOed synthetic data set, with missing traces, and (b) reconstructed result.

When we address the problem of interpolation beyond aliasing, either we should introduce some kind of model styling to exploit the information we have about the model, or we should generate some kind of proxy data. In this article, I briefly discuss three such methods of generating the proxy data that have imperfect, but still promising results, and I present results for one of them. Using the techniques presented below, I create a proxy data onto which I estimate the covariance-based filter and then interpolate the original data.

1. The crudest way of making the covariance estimator pick the trends we want is to generate a proxy data by linear interpolation along the known dips. The method first estimates the dips on the aliased data, and then does linear interpolation along those dips. Once we have the results from linear interpolation, we can estimate our covariance-based interpolation filter on this data, and then repeat the process of interpolation with the estimated filter. The results obtained from this technique were good, but this method has a severe shortcoming: it restricts the solution to a finite number of choices in terms of dips, and in places with conflicting dips, this might enhance one dip relative to the other. On certain occasions, low-energy events might be of interest, and in the presence of high-energy events, the dip estimator might miss the low-energy events, eliminating them from the solution. As a result, in spite of promising first results I was discouraged from using this method.

2. A second, more sophisticated approach is to use the pyramid-domain representation of the aliased data (Burt and Adelson, 1983; Sen, 2006). It is a low-frequency representation on higher-level pyramids and can potentially be useful for removing aliasing arti-

facts. Initial results with this method were not good, but with more work, this technique might be exploited to handle problems of aliasing.

3. The last approach presented here is that of creating envelopes to arrive at a low-frequency unaliased representation of the original data. The standard approach is to take the Hilbert transform, and then estimate the absolute value (Claerbout, 1992). Since that would be an extremely low-frequency representation, inadequate for estimation of covariances, I take the absolute value without taking the Hilbert transform. Strictly speaking, this is not the envelope, but an intermediate-frequency representation rather than a low-frequency one. Figure 15(a) shows the envelope (absolute value), and Figure 15(b) is interpolated result. As can be noticed, results are not perfect but much better than results shown in figure 9(c)

Figure 15: (a)Psuedo-envelope for hask data, and (b) results obtained with envelopes.

3D RESULTS FROM GULF OF MEXICO

Finally, I test the proposed method on a real 3D DMO stacked data set acquired by UNOCAL in Gulf of Mexico. The first step is to look at a few 2D lines from within the 3D data cube. The sampling interval along both inline and crossline directions is 110 ft. Since our final goal is to achieve a high-resolution image, I infill the inline section with zero traces and then interpolate. A window from one inline section before and after interpolation is given in Figure 16. In general, the result looks good and is of a higher resolution. But, in another window at a finer scale taken from same line (Figure 17), we observe a discrepancy between the interpolated and the observed amplitude (a blocky pattern). This happens primarily because in real situations things are not perfect; we record noise as well as signal, and the strength of the signal is not uniform everywhere. As a result, our estimate of covariance includes characteristics of both

signal and noise. Therefore, some sort of regularization or model styling would be desirable while interpolating, to ensure a certain degree of smoothness in the final image.

The implementation of the proposed interpolation filter in 3D is similar to that in 2D, the only difference being that in 2D we considered four diagonal neighbors situated at corners of a square for interpolation, whereas in 3D we consider 8 diagonal neighbors situated at corners of a cube. Fortunately, this data required no anti-aliasing technique, and the results shown were achieved with the method in its original form.

As part of interpolation, I re-sample both the inline and crossline axis by half. Interpolation was carried out first in the inline direction and then in the crossline direction. This approach is equivalent to estimating both at once; infact they are computationally different but conceptually the same. Since we are interpolating along both inline and crossline directions, depth (time) slices will be the best places to look for the differences. In Figure 18, I compare the time slices before and after interpolation taken at 3 seconds. Similarly a comparison between slices taken at 4 seconds is drawn in Figure 19. Apart from resolution enhancement in the interpolated time slice, we also notice some patching artifacts (linear, grid-like features). Patching was introduced to maintain the assumption of local stationarity before applying the algorithm to larger dimensions. The artifacts creep in, because when we have an unknown trace at the end of a patch, we do not have non-zero diagonal neighbors on both the sides, so we need to make an approximation about the boundary. For this example I assumed *Zero-Slope Boundary Conditions*.

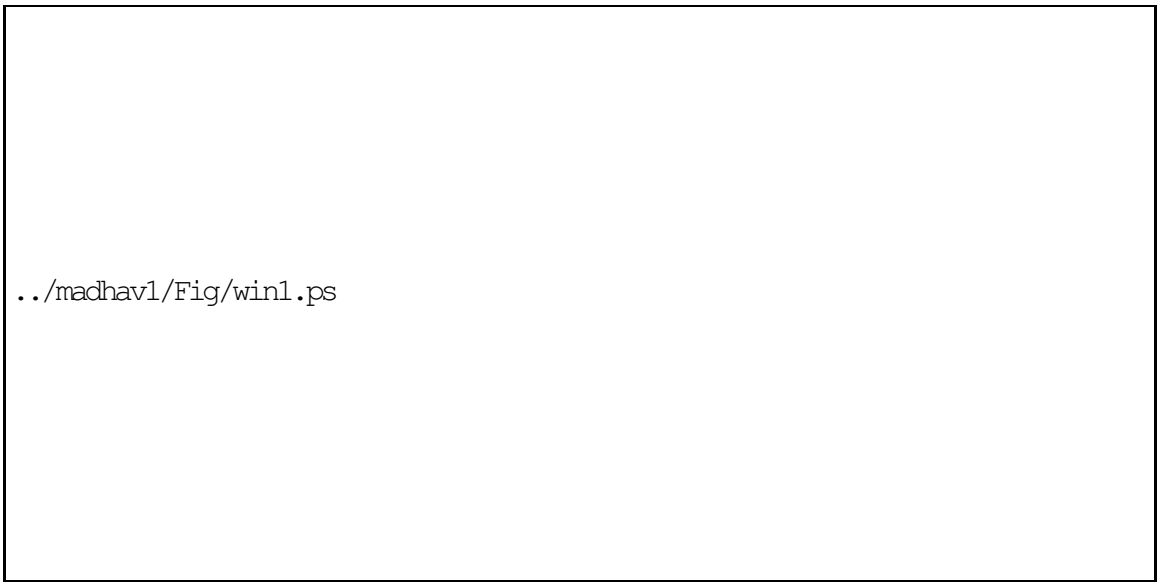

Figure 16: Window from an inline section; panels show original (left) and interpolated (right) data.

The main reason for increasing the resolution of this data set was to see whether we could improve the migration results. Figure 20 shows a cubeplot taken at the center of the migrated cube before interpolation, and Figure 21 shows migrated results at same location. In the latter case, migration was carried out after interpolation. The migrated results created after

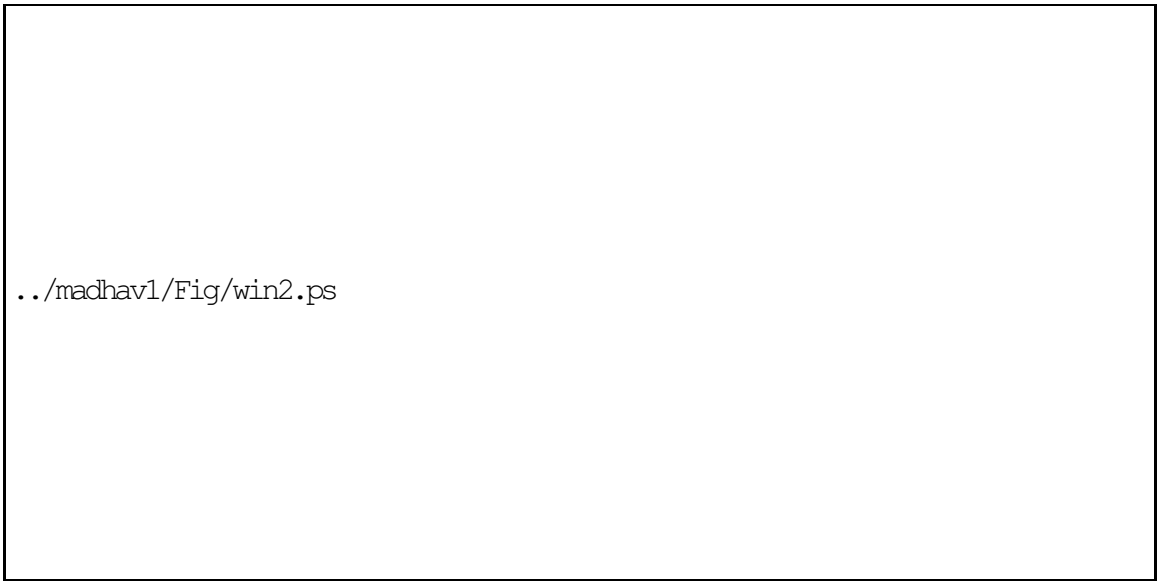

Figure 17: Window from an inline section before (left) and after (right) resolution enhancement demonstrating blocky pattern in the result.

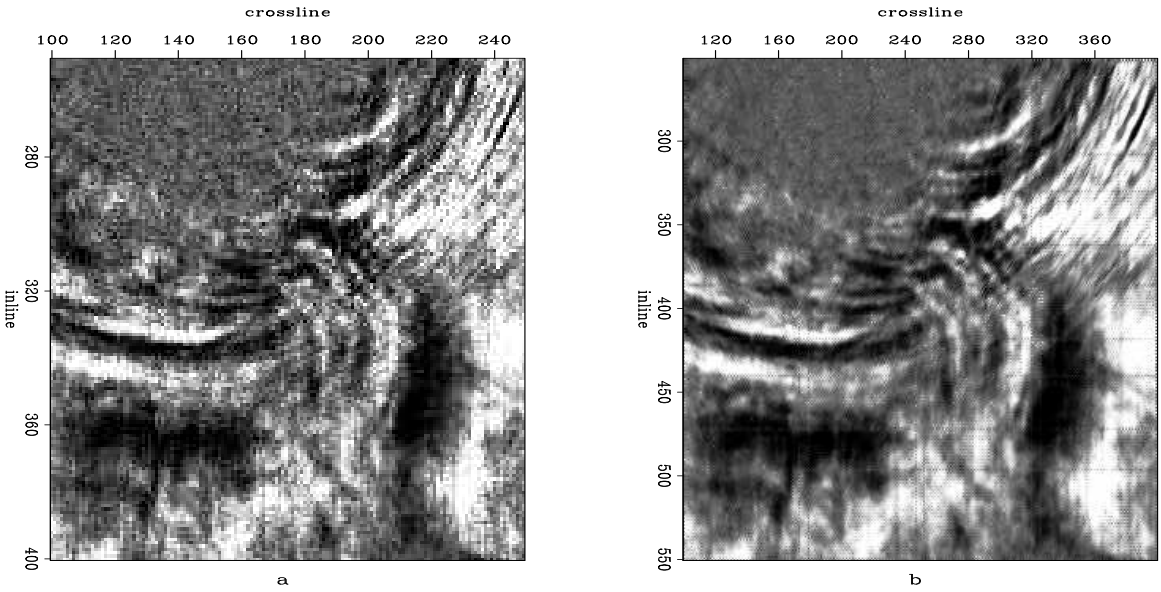

Figure 18: Time slice taken at 3 seconds (a) before interpolation and (b) after interpolation. madhav1-slice1f [CR]

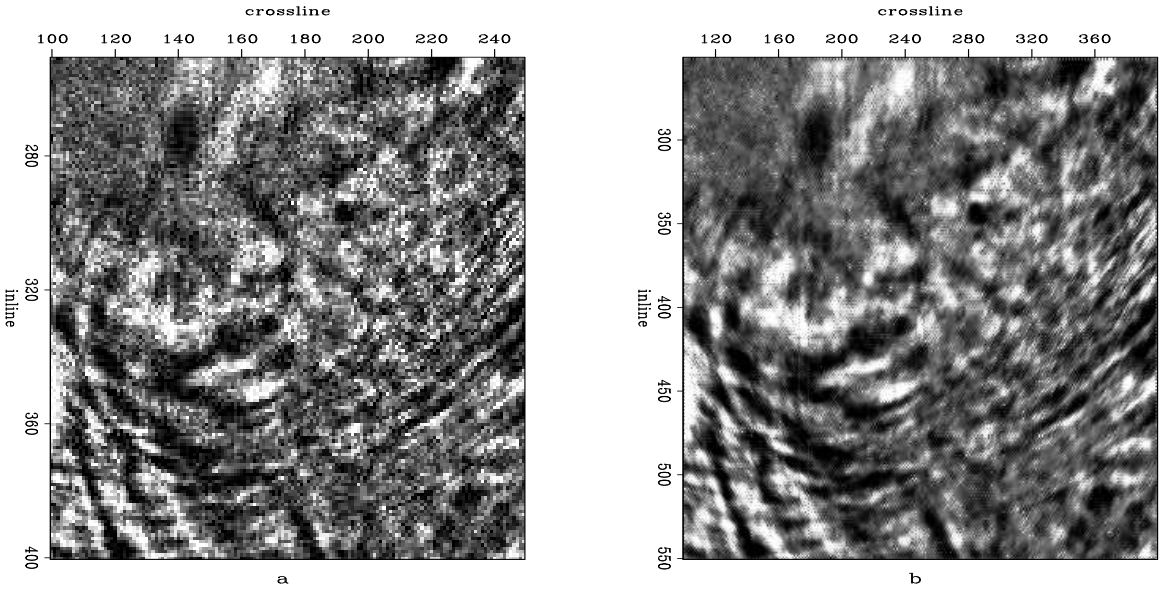

Figure 19: Time slice taken at 4 seconds (a) before interpolation and (b) after interpolation. madhav1-slice2f [CR]

interpolation look marginally better than the ones created without. The difference is a little more pronounced when we window a small portion and look closely, as displayed in Figure 22. On one hand there is an obvious enhancement in the resolution but on the other, there is a substantial increase in the computational costs, since the data volume after interpolation was four times larger. Finally let us compare a depth slice taken from the image cube at 6000 ft, with migration carried out before (Figure 23(a)) and after interpolation (Figure 23(b)). Results obtained by migration after interpolation look particularly better inside the salt structure and on top right corner where some channels come out more clearly.

We can conclude that the proposed covariance-based interpolation scheme worked reasonably well for the 3D data set under considertaion. The algorithm in its present form can handle slight aliasing, but it would be interesting to experiment with severly aliased data sets.

REGULARIZATION

Some of the results have shown that there is an amplitude discrepancy between the interpolated and the actual data values. Results of simple energy-balancing conditions were not satisfactory. Hence, I propose a modification in the interpolation scheme to obtain smoother results from real data sets without compromising the sharpness and resolution. In its original form, the method uses the filter to fill in the missing values in a single step. If we instead introduce an inversion goal similar to the one given in the following equations, we will get much smoother

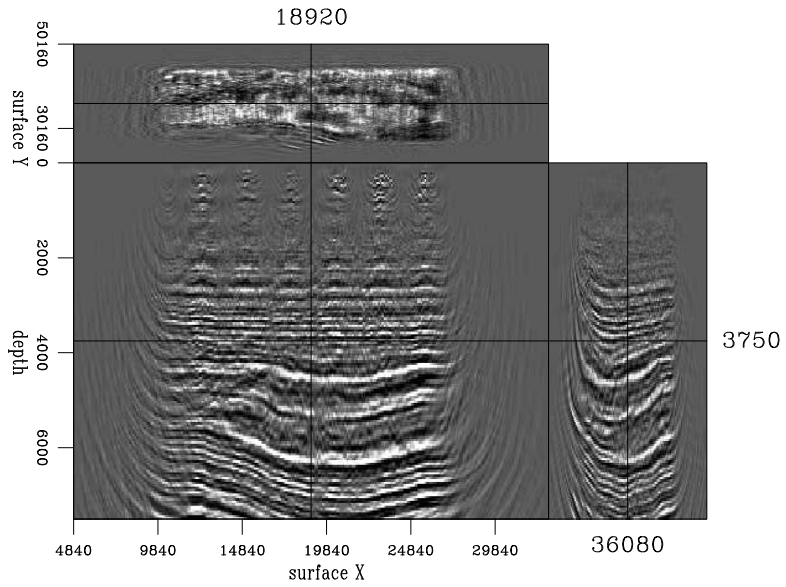

Figure 20: Cubeplot taken at the center of the migrated cube before interpolation. madhav1-3face [CR]

results at the cost of extra computation time.

$$(\mathbf{d} - \mathbf{Km}) \approx \mathbf{0} \tag{9}$$

$$\epsilon \mathbf{Am} \approx \mathbf{0}. \tag{10}$$

Here, **K** is the mask for known values, **A** is the filter determined from the covariance structure, **d** is the known data, and **m** is the smooth interpolated model that we desire. Figure 24 draws a comparison between the interpolated and the regularized result. Notice that the blocky pattern present in the interpolated image is no longer present in the regularized result.

CONCLUSIONS

The proposed method has shown mixed results. The assumption of local stationarity has far-reaching implications if we estimate the covariance structure and the filter coefficients on an image made up of random features; the method's performance in this case would be similar to fourth order linear interpolation. This happens because the covariance structure of a fairly random image has no directional preference to tune the coefficients, so it takes equal contribution from all directions. On the other hand, if an image has lines dipping in one particular direction, covariance would be lower along the dip than across the dip, and filter coefficients would be tuned accordingly.

Therefore, ideally we would want to divide our image into as many patches as possible and then estimate the covariance structure and filter coefficients of each patch separately, but there are trade-offs. First, the computation cost would go up in direct proportion to the number of patches. More importantly, in small patches the covariance structure is more likely to be

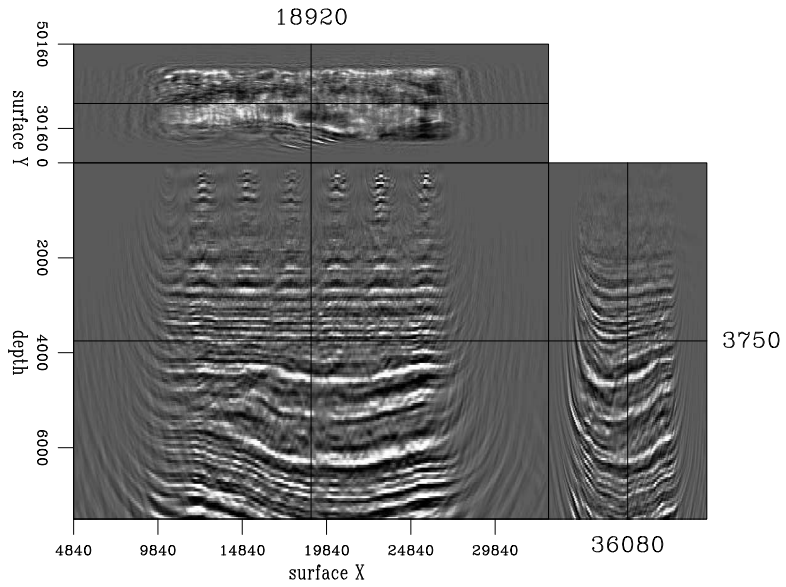

Figure 21: Cubeplot taken at the center of the migrated cube after interpolation. madhav1-interp.3face [CR]

affected by local noise rather than global features. In the case of real data examples, where some noise was present, it was observed that use of regularized inversion gave better results.

Another potential application of the covariance-based interpolation filter is to use this filter as a roughener for regularization. This would ensure smoothing not along the cartesian mesh but along edges and discontinuities present in the model, without requiring to explicitly determine them. Though, computing inverse of this filter might not be straight forward. Clapp et al. (1997) discusses the idea of a steering filter where we can steer along the known or estimated geological dips, but with a covariance-based approach, we can achieve the same goal without estimating any dip explicitly, instead using the covariance structure to quantify the same.

ACKNOWLEDGMENTS

I thank UNOCAL for the data and Jeff Shragge for many insightful discussions and suggestions.

REFERENCES

Burt, P. J. and E. H. Adelson, 1983, The Laplacian Pyramid as a Compact Image Code: IEEE Transactions on Communications, **31**, 532–540.

Claerbout, J. F., 1992, Earth Soundings Analysis: Processing versus inversion: Class notes, http://sepwww.stanford.edu/sep/prof/index.html.

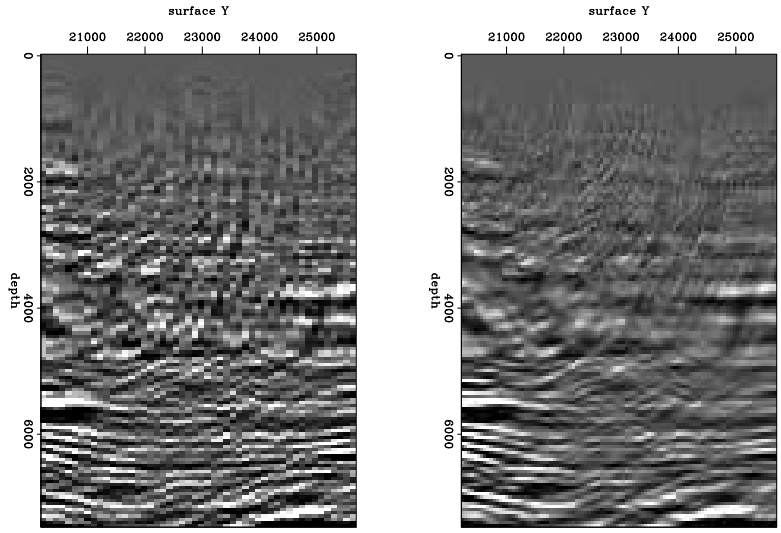

Figure 22: Window from within the migrated result, (a) without interpolation and (b) with interpolation. madhav1-migwin [CR]

Claerbout, J. F., 2005, Image Estimation by Example: Class notes, http://sepwww.stanford.edu/sep/prof/index.html.

Clapp, R. G., S. Fomel, and J. Claerbout, 1997, Solution steering with space-variant filters: SEP–**95**, 27–42.

Ji, J. and J. F. Claerbout, 1991, Trace interpolation using recursive dip filters: SEP–**72**, 43–56.

Li, X. and M. T. Orchard, 2001, New edge-directed interpolation: IEEE Transactions on Image Processing, **10**, 1521–1527.

Sen, S., 2006, Missing data interpolation with gaussian pyramids: SEP–**124**.

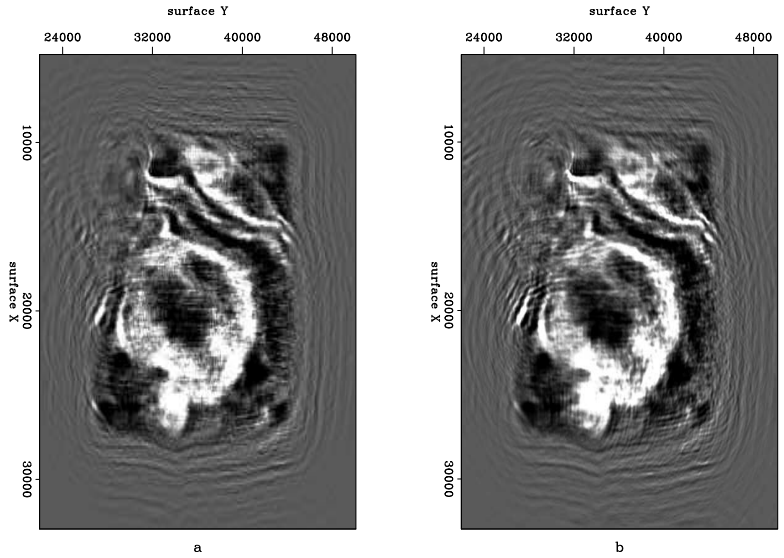

Figure 23: Depth slice taken at 6000 ft from migrated cube (a) without interpolation and (b) with interpolation. madhav1-migslicef [CR]

../madhav1/Fig/win3.ps

Figure 24: Window from inline section comparing interpolated result with (right) and without (left) regularization.