

Dynamic warping of seismic images

Dave Hale

Center for Wave Phenomena, Colorado School of Mines, Golden CO 80401, USA

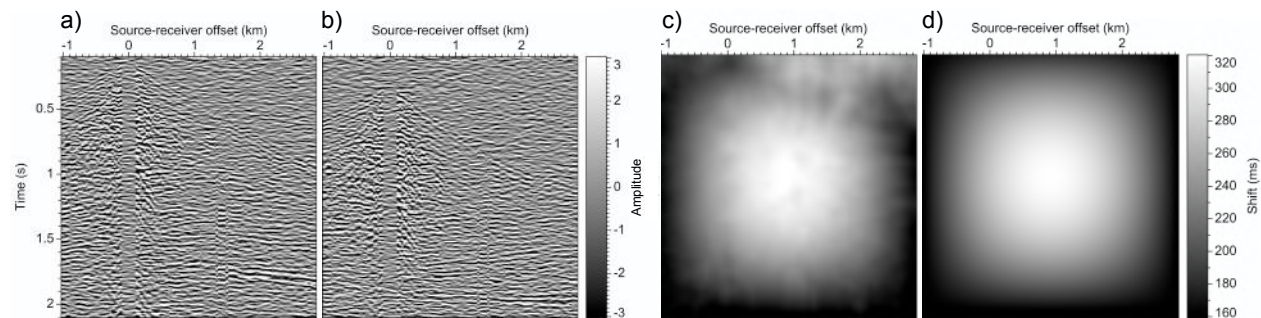


Figure 1. A recorded seismic shot record before (a) and after (b) warping with large shifts that vary with time and offset. Reflections are obscured by two different ambient bandlimited noise images that have been added to the original and warped shot records. (The rms signal:noise ratio in this example is 1:1.) Shifts (c) computed from these two images by a dynamic warping algorithm approximate well the known shifts (d) used to perform the actual warping.

ABSTRACT

The problem of estimating relative time (or depth) shifts between two seismic images is ubiquitous in seismic data processing. This problem is especially difficult where shifts are large and vary rapidly with time and space, and where images are contaminated with noise. I propose a solution to this problem that is a simple extension of the classic dynamic time warping algorithm for speech recognition. This new *dynamic image warping* method for estimating shifts is more accurate than methods based on crosscorrelation of windowed images where shifts vary significantly within the windows.

Key words: seismic image dynamic warping correlation

1 INTRODUCTION

In seismic data processing we must often estimate relative shifts in time (or depth) between seismograms. Often those shifts vary with both time and space coordinates. Examples cited by Liner and Clapp (2004) include alignment of synthetic and recorded seismograms, registration of P- and S-wave images, residual normal moveout correction, and alignment of images computed for different source-receiver offsets or propagation angles. They proposed a dynamic programming solution to this problem in the case where pairwise alignment between seismic traces is sufficient.

A different dynamic programming solution was developed by Sakoe and Chiba (1978) in the context of speech recognition, and is today widely known as *dy-*

namic time warping (DTW; e.g., Müller, 2007, Chapter 4). Significantly, DTW imposes constraints on the rate at which shifts may vary in time, and these constraints often enable DTW to accurately estimate shifts from sequences that are contaminated with noise, or that in some other ways are not simply warped versions of each other.

The use of dynamic time warping to estimate shifts in geophysical time series and other sequences is not new. Several applications of dynamic time warping to problems in geophysics were proposed by Anderson and Gaby (1983), who called this algorithm “dynamic waveform matching”.

Unfortunately, the most straightforward extension of DTW to the problem of estimating shifts in multi-dimensional images has been shown to be NP-complete

(Keyzers and Unger, 2003), that is, computationally intractable. Therefore, numerous authors — Pishchulin (2010) provides a recent summary — have proposed practical solutions to problems that approximate this NP-complete problem.

In this paper I propose an extension of an approximate solution developed by Mottl et al. (2002). Their solution and my extension for *dynamic image warping* (DIW) are especially simple, requiring very little software beyond what would already be available to implement dynamic time warping. I provide the essential computer software components for both DTW and DIW in Appendix A.

I first review the dynamic time warping algorithm, giving special attention to the so-called accumulation and backtracking parts of this algorithm. I then show how the accumulation part of DTW can be used to implement a non-linear smoothing of alignment errors computed for two multi-dimensional images, and how this leads to a new method for DIW.

In tests with pairs of images related by shifts that are known, large, and rapidly varying, I demonstrate the accuracy with which DIW can estimate the known shifts. Figure 1 is an example of one such test for two images contaminated with bandlimited random noise.

In further tests I show that DIW can be more accurate than methods based on local crosscorrelations, especially when shifts vary rapidly in time or space. Crosscorrelation methods, such as those proposed by Hall (2006) and Hale (2009) to estimate shifts in time-lapse seismic images, are accurate only where shifts are more slowly varying.

2 DYNAMIC TIME WARPING

Consider the two synthetic seismograms $f[i]$ and $g[i]$ with length $N = 500$ samples displayed in Figure 2. I computed the sequence $f[i]$ by convolving a Ricker wavelet with a random reflectivity sequence. I then applied time-varying shifts $s[i]$ to that reflectivity sequence and convolved again with the same wavelet to obtain the sequence $g[i]$. The two sequences are therefore approximately (not exactly) related by $f[i] \approx g[i + s[i]]$.

In practice these two sequences might be a recorded seismogram $f[i]$ and a synthetic seismogram $g[i]$ derived from well logs. Or they might be two sequences of sample values extracted from a seismic image on opposite sides of a fault. In any case, the practical problem considered here is estimation of the shifts $s[i]$ given only the two sequences $f[i]$ and $g[i]$.

In the example of Figure 2, the shifts $s[i]$ are a simple sinusoidal function that is apparent in Figure 3a, which is an image of alignment errors defined by

$$e[i, l] \equiv (f[i] - g[i + l])^2. \quad (1)$$

Note that these alignment errors are nearly zero where the integer lag l approximately equals the shift $s[i]$. Also

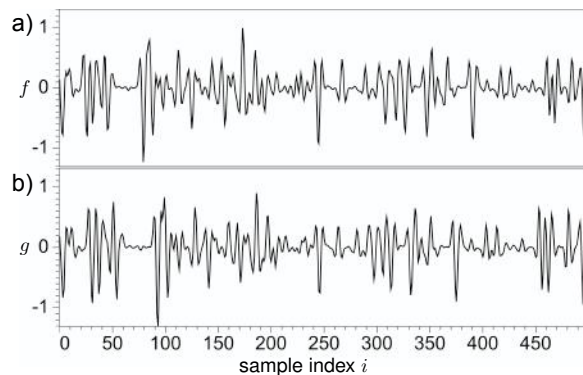


Figure 2. Two synthetic seismograms $f[i]$ (a) and $g[i]$ (b) corresponding to misaligned reflection coefficients, used as inputs to the dynamic time warping algorithm. Reflections in $f[i]$ also appear in $g[i]$, but are squeezed toward the middle of the sequence.

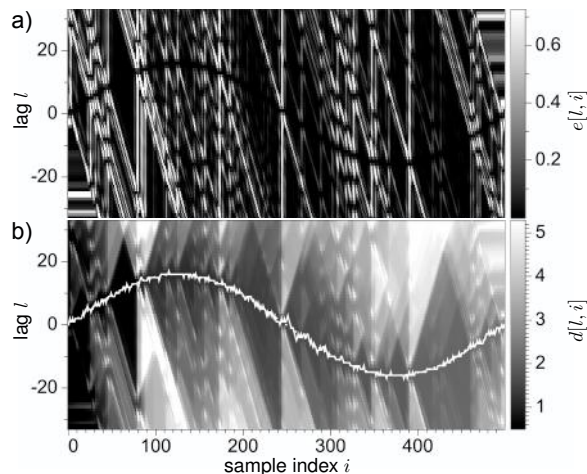


Figure 3. Alignment errors $e[i, l]$ (a) are small along the sinusoidal path corresponding to known shifts between reflections in the two sequences shown in Figure 2. Dynamic time warping (b) yields (solid white) estimated integer shifts $u[i]$ that approximate well the (dotted white) known shifts $s[i]$ in reflection coefficients.

note the constant extrapolation of errors in the corners of $e[i, l]$, where $i + l < 0$ or $i + l \geq N$.

The definition of alignment errors in equation 1 can be modified by changing the power 2 or by using some other non-negative function of the differences $f[i] - g[i + l]$, without changing the dynamic time warping algorithm. For example, we might use the absolute values of those differences. In all of the examples shown in this paper I have simply squared the differences, as in equation 1.

2.1 Constrained optimization

The simplest dynamic time warping (DTW) computes a sequence $u[0 : N - 1] \equiv (u[0], u[1], \dots, u[N - 1])$ of integer shifts by solving the following optimization problem:

$$u[0 : N - 1] = \arg \min_{l[0 : N - 1]} D(l[0 : N - 1]), \quad (2)$$

where

$$D(l[0 : N - 1]) \equiv \sum_{i=0}^{N-1} e[i, l[i]], \quad (3)$$

subject to the constraint

$$|u[i] - u[i - 1]| \leq 1. \quad (4)$$

As illustrated in Figure 3b, DTW yields a minimizing sequence of integer shifts $u[0 : N - 1]$ that well approximates the (here, known) sequence of shifts $s[0 : N - 1]$.

The function D defined by equation 3 is often referred to as *distance*, which makes sense if we think of the image $e[i, l]$ in Figure 3a as representing some topography. Large values in $e[i, l]$ then correspond to tall hills (misalignments) that we wish to avoid as we choose a path from left to right, that is, from $i = 0$ to $N - 1$. In this sense, DTW chooses a path $u[0 : N - 1]$ to minimize the total distance traveled, subject to the constraint (equation 4) that the shifts cannot change too rapidly from one sample to the next.

The constraint equation 4 is analogous to the simplest slope constraint of Sakoe and Chiba (1978). This constraint is slightly different here only because I define alignment errors by $e[i, l] \equiv (f[i] - g[i + l])^2$ (equation 1), instead of by $e[i, j] \equiv (f[i] - g[j])^2$. Here this constraint ensures that the argument $i + u[i]$ in $(f[i] - g[i + u[i]])^2$ neither decreases nor increases too rapidly with increasing sample index i .

This constraint is important. Where $u[i] - u[i - 1] = 1$, we stretch by 100%, such that two adjacent samples in the sequence $f[i]$ correspond to two non-adjacent samples in the sequence $g[i]$. Where $u[i] - u[i - 1] = -1$, we squeeze by 100%, such that two adjacent samples in the sequence $f[i]$ correspond to only one sample in the sequence $g[i]$. In many practical applications 100% is an unreasonably large amount of strain, and we will see below how to reduce this upper bound.

It is significant that the sequence $u[0 : N - 1]$ computed by DTW minimizes exactly the distance D defined by equation 3, while satisfying the constraint equation 4. Differences in Figure 3b between the integer shifts $u[0 : N - 1]$ and the known shifts $s[0 : N - 1]$ are due entirely to the restriction of $u[0 : N - 1]$ to be integers and the approximation $f[i] \approx g[i + s[i]]$, not to any approximation in the optimization algorithm.

2.2 Dynamic programming

As its name implies, dynamic time warping is a dynamic-programming algorithm (e.g., Cormen et al.,

2001). The essential trait of this algorithm is decomposition of a problem into a sequence of nested and smaller subproblems.

Let $u[0 : N - 1]$ denote the sequence of shifts l that minimizes the distance D defined by equation 3. To identify the sequence of smaller subproblems nested within this larger minimization problem, we consider a subpath $u[0 : m]$ of the minimizing path $u[0 : N - 1]$ and observe that

$$u[0 : m] = \arg \min_{l[0 : m]} \sum_{i=0}^m e[i, l[i]]. \quad (5)$$

For if $u[0 : m]$ were not a minimizing subpath, then we could replace that part of $u[0 : N - 1]$ and thereby reduce the total distance D , which implies that $u[0 : N - 1]$ does not minimize D , a contradiction.

This observation is important because it implies that we need not test all possible (roughly, 3^N) paths $l[0 : N - 1]$ that satisfy the constraint equation 4 in our search for the minimizing path $u[0 : N - 1]$. Instead, we can find this minimizing path in two steps: accumulation and backtracking.

2.3 Accumulation

In the first accumulation step of DTW, we recursively compute from the array of alignment errors $e[i, l]$ an array of distances $d[i, l]$ as follows:

$$\begin{aligned} d[0, l] &= e[0, l], \\ d[i, l] &= e[i, l] + \min \begin{cases} d[i - 1, l - 1] \\ d[i - 1, l] \\ d[i - 1, l + 1] \end{cases}, \\ &\text{for } i = 1, 2, \dots, N - 1. \end{aligned} \quad (6)$$

For each index i , we cannot yet know in this first step whether or not the lag l lies on the minimizing path $u[0 : N - 1]$, so that $l = u[i]$. Therefore, we must compute and store distances $d[i, l]$ for all lags, assuming for the moment that lag l may lie on the minimizing path. Figure 3b shows distances $d[i, l]$ computed in this way for the alignment errors shown in Figure 3a.

The constraint equation 4 implies that, when computing $d[i, l]$ as in equation 6, we must consider only three previously computed distances $d[i - 1, l - 1]$, $d[i - 1, l]$, and $d[i - 1, l + 1]$. In other words, if lag l lies on the minimizing path at sample index i , then either lag $l - 1$, l or $l + 1$ must lie on the minimizing path at sample index $i - 1$.

The computational complexity of this first step is $O(N \times L)$, where L is the number of lags l for which alignment errors $e[i, l]$ and distances $d[i, l]$ are computed.

I call this first step *accumulation* because the distances $d[i, l]$ are sums of alignment errors $e[i, l]$. At the end of this first step, we can simply loop over all lags l

to find the minimum distance

$$D = \min_l d[N-1, l]. \quad (7)$$

2.4 Backtracking

The second step in DTW is to find the minimizing path, the sequence of shifts $u[0 : N-1]$, beginning with the last shift $u[N-1]$ and ending with the first shift $u[0]$:

$$\begin{aligned} u[N-1] &= \arg \min_l d[N-1, l], \\ u[i-1] &= \arg \min_{l \in \{u[i]-1, u[i], u[i]+1\}} d[i-1, l], \\ &\text{for } i = N-1, N-2, \dots, 1. \end{aligned} \quad (8)$$

This *backtracking* step begins with a simple loop over lags l to find the last shift $u[N-1]$ in the sequence of shifts $u[0 : N-1]$. Because this last shift must be on the minimizing path, it must equal the lag at which we found the minimum distance D .

We then recursively find previous shifts $u[i-1]$ in this sequence, comparing the three distances $d[i-1, l-1]$, $d[i-1, l]$ and $d[i-1, l+1]$ to determine which of these was used in equation 6 to compute the minimum distance $d[i, l]$.

The computational complexity of backtracking is only $O(N)$, because for each sample index i we compute a shift $u[i]$ by comparing only three distances. Therefore, the complexity of DTW is $O(N \times L)$, that of the accumulation step, which is proportional to the number of samples in the image of alignment errors $e[i, l]$.

3 REFINEMENTS

Figure 4 displays the same two synthetic seismograms $f[i]$ and $g[i]$ shown in Figure 2, after adding different sequences of bandlimited random noise to each of them. The rms signal:noise ratio is 2:1. While this level of noise obscures somewhat the sinusoidal warping path in the plot of alignment errors $e[i, l]$ in Figure 5a, the shifts $u[i]$ estimated using DTW are a rough approximation to the known shifts $s[i]$.

Again it is important to remember that DTW solves exactly the constrained optimization problem of equations 2–4. Differences in Figure 5b between estimated and known shifts are primarily due to errors in the approximation $f[i] \approx g[i + s[i]]$ caused by the addition of random noise. The sequence $g[i]$ in Figure 4b is not simply a warped version of the sequence $f[i]$ in Figure 4a.

3.1 Limiting strain

The robustness of DTW in the presence of random noise is due largely to the constraint equation 4. The number

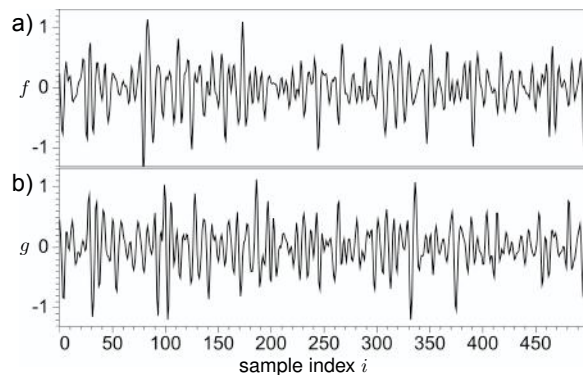


Figure 4. Same as Figure 2, except that bandlimited random noise sequences have been added to the synthetic seismograms $f[i]$ (a) and $g[i]$ (b). In this example the rms signal:noise ratio is 2:1.

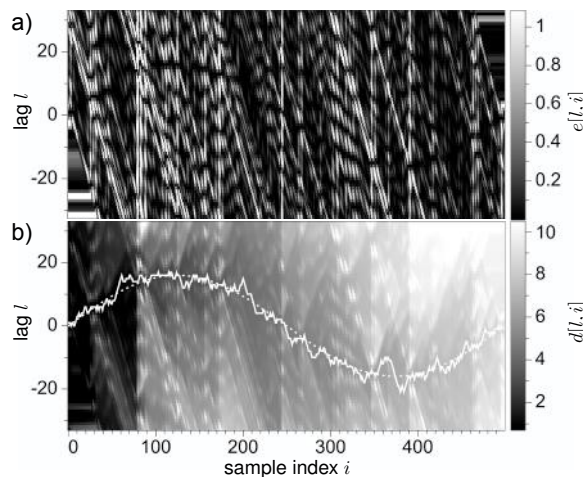


Figure 5. Known sinusoidal warping is obscured in alignment errors $e[i, l]$ (a) computed for the noisy synthetic seismograms of Figure 4. Dynamic time warping (b) yields (solid white) estimated integer shifts $u[i]$ that roughly approximate the (dotted white) known shifts $s[i]$ in reflection coefficients.

of shift sequences $u[0 : N-1]$ that satisfy this constraint ($\approx 3^N$) is far less than the number that would be possible without it ($\approx L^N$).

Of course, the constraint that strain (stretch or squeeze) be less than 100% is useful only when satisfied by the actual shifts $s[i]$ that we wish to estimate. However, in many practical applications this constraint is more than reasonable. Indeed, a strain as high as 100% may be unreasonably high, and we may be able to improve the accuracy of shifts estimated in DTW by reducing this upper bound on strain to a more reasonable value.

The simplest way to more tightly bound strain in DTW is to sample lags l more finely at some fraction of the time sampling interval. For example, if that frac-

tion were $\frac{1}{2}$, then we would compute alignment errors $e[i, l]$ for lags $l = \dots, -1, -\frac{1}{2}, 0, \frac{1}{2}, 1, \dots$. The maximum strain permitted would then be 50%, as the constraint equation 4 would become

$$|u[i] - u[i - 1]| \leq \frac{1}{2} \quad (9)$$

While straightforward, this method for reducing the upper bound on strain requires a significant increase in computational cost. For a limit of 50%, both computation time and memory will double if we compute errors $e[i, l]$ and distances $d[i, l]$ for twice as many lags. The increase in memory will be especially significant as we extend the dynamic time warping algorithm to the problem of multi-dimensional image warping.

A more efficient way to limit strain is to implement constraints much like the slope constraints proposed by Sakoe and Chiba (1978). As an example, for a limit of 50% strain, I change the accumulation step (equation 6) to compute distances as

$$\begin{aligned} d[0, l] &= e[0, l], \\ d[1, l] &= e[1, l] + \min \begin{cases} d[0, l - 1] \\ d[0, l] \\ d[0, l + 1] \end{cases}, \\ d[i, l] &= e[i, l] + \min \begin{cases} d[i - 2, l - 1] + e[i - 1, l - 1] \\ d[i - 1, l] \\ d[i - 2, l + 1] + e[i - 1, l + 1] \end{cases}, \end{aligned} \quad (10)$$

for $i = 2, 3, \dots, N - 1$.

A corresponding change is required in the backtracking step, in which we must now compute and compare the three expressions inside the min function of equation 10, to determine which of these was used to compute the distance $d[i, l]$.

The effect of these modifications is to impose the following constraint on changes in shifts:

$$|u[i] - u[i - 1]| + |u[i - 1] - u[i - 2]| \leq 1. \quad (11)$$

Like equation 9, this equation is similar to (but not quite equivalent to) a finite-difference approximation to $|du/dt| \leq \frac{1}{2}$.

In words, dynamic time warping based on equation 10 is constrained to shift sequences in blocks of two or more samples. If any sample is shifted by the warping, then at least one of the adjacent samples must be shifted by the same amount.

Modifications similar to equation 10 can be easily and efficiently implemented for any strain limit of the form $1/b$, where b is an integer. (See Appendix A.) Figure 6 shows how strain limits implemented in this way can improve the accuracy of shifts estimated by DTW.

Note, however, that the strain limit of $\frac{1}{5}$ used to estimate shifts $u[i]$ shown in Figure 6c is almost equal to the maximum strain in the known shifts $s[i]$. Any further reduction in the strain limit would yield poor shift estimates, because strain for the correct shifts would

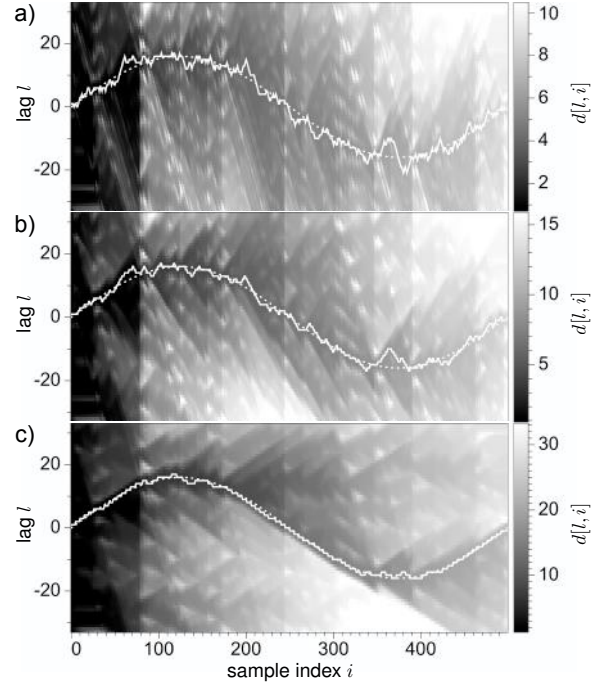


Figure 6. Shifts $u[i]$ estimated by dynamic time warping for different limits on strain, the rate at which shifts can change with sample index i . As we reduce the upper bound on this strain from 1 (a) to $\frac{1}{2}$ (b) to $\frac{1}{5}$ (c), the (solid white) estimated shifts $u[i]$ better approximate the (dotted white) known shifts $s[i]$.

exceed that limit. In practice, lacking any a priori limit on strain, we must take care to not reduce this limit so much that we prohibit the correct shifts.

3.2 Smoothing alignment errors

To further improve the accuracy of estimated shifts $u[i]$, we might attempt to attenuate noise in the two sequences $f[i]$ and $g[i]$, or we might instead try to attenuate noise in the alignment errors $e[i, l]$. Considering the second option, suppose that we apply some sort of smoothing filter to the alignment errors $e[i, l]$. Can we improve the accuracy of the estimated shifts $u[i]$ by applying such a filter before DTW?

This question is suggested by the accumulation step in DTW defined by equation 6. Each distance $d[i, l]$ computed in this step is a sum of alignment errors, which implies that the distances $d[i, l]$ vary less rapidly with index i than do the alignment errors $e[i, l]$. In other words, the accumulation step is a smoothing filter.

This recursive smoothing filter is one-sided, because each $d[i, l]$ in equation 6 depends on only previous and present alignment errors, those with sample indices less than or equal to i . This filter is also non-linear, because of the min function in equation 6. In effect, this one-sided non-linear smoothing filter already attenuates

noise in alignment errors caused by noise in the two sequences to be aligned by warping.

One way we might improve this smoothing filter would be to make it two-sided and symmetric. We can implement a two-sided symmetric smoothing filter by applying a one-sided filter in forward and reverse directions. Smoothing in the forward direction is the same as computing distances $d[i, l]$ via equation 6:

$$\begin{aligned} \tilde{e}_f[0, l] &= e[0, l], \\ \tilde{e}_f[i, l] &= e[i, l] + \min \begin{cases} \tilde{e}_f[i-1, l-1] \\ \tilde{e}_f[i-1, l] \\ \tilde{e}_f[i-1, l+1] \end{cases}, \end{aligned} \quad (12)$$

for $i = 1, 2, \dots, N-1$.

Smoothing in the reverse direction is similar:

$$\begin{aligned} \tilde{e}_r[N-1, l] &= e[N-1, l], \\ \tilde{e}_r[i, l] &= e[i, l] + \min \begin{cases} \tilde{e}_r[i+1, l-1] \\ \tilde{e}_r[i+1, l] \\ \tilde{e}_r[i+1, l+1] \end{cases}, \end{aligned} \quad (13)$$

for $i = N-2, N-3, \dots, 0$.

Two-sided smoothing is then defined by

$$\tilde{e}[i, l] = \tilde{e}_f[i, l] + \tilde{e}_r[i, l] - e[i, l]. \quad (14)$$

Subtraction of $e[i, l]$ in equation 14 ensures that this value is not counted twice, as for all i it appears in both $\tilde{e}_f[i, l]$ and $\tilde{e}_r[i, l]$. In this way, each smoothed error $\tilde{e}[i, l]$ is a sum of past, present and future alignment errors.

Like the accumulator in DTW, this two-sided smoothing filter is non-linear because it uses the min function in both equations 12 and 13 to determine which errors to sum. Figure 7 displays smoothed alignment errors for the two noisy sequences shown in Figure 4.

Observe that the known sinusoidal warping path is somewhat more apparent in these smoothed errors than in the unsmoothed alignment errors displayed in Figure 5a. We might therefore expect the shifts $u[i]$ estimated by DTW from the smoothed errors $\tilde{e}[i, l]$ would be more accurate than those estimated from the unsmoothed errors $e[i, l]$.

However, this sort of smoothing does not improve DTW. Although not shown here, the shifts $u[i]$ computed by DTW for the smoothed alignment errors shown in Figure 7c are identical to those computed for the unsmoothed alignment errors in Figure 5a. The benefit of this two-sided smoothing lies in the extension of dynamic warping for one-dimensional sequences to that for multi-dimensional images.

4 DYNAMIC IMAGE WARPING

The simplest way to extend dynamic time warping for image processing is to think of an image as a collection of vertical columns and to estimate vertical shifts by

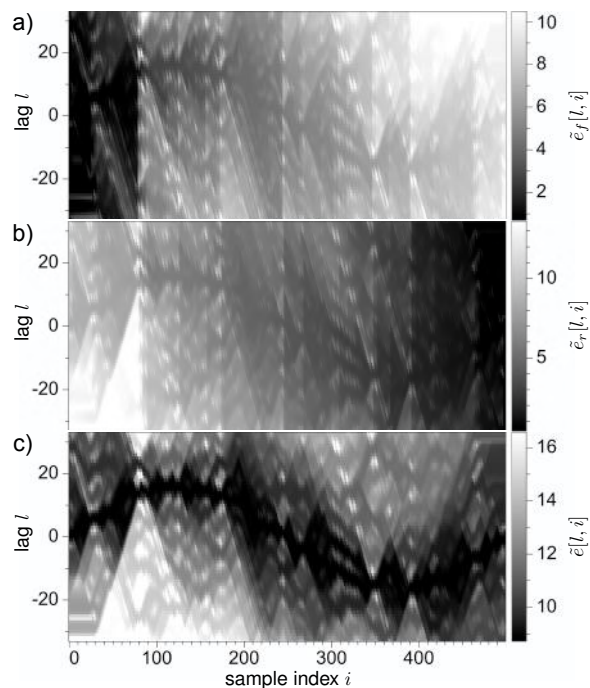


Figure 7. Alignment errors for the noisy sequences in Figure 4 after smoothing in the forward (a), reverse (b), and both directions (c). Smoothing in the forward direction is equivalent to the accumulation step in DTW, in which we compute distances $d[i, l]$ via equation 6.

applying DTW to each of those columns independently. We could likewise apply DTW to image rows to obtain estimates of horizontal shifts.

Figure 8 illustrates the application of this simple method for dynamic image warping for two seismic shot records, where the first record shown in Figure 8a has been warped to obtain the second record shown in Figure 8b. DTW applied to each corresponding pair of columns from these images yields the estimated shifts shown in Figure 8a. Except for small source-receiver offsets where seismograms are missing, the estimated shifts approximate well the known shifts used to warp the images.

These shifts are large, about eight times larger than the dominant period of most reflection events, which is about 40 ms. And many of the events in the shot records (such as those for small offsets and late times) are ringy, almost periodic, which can make estimation of the shifts more difficult. Nevertheless, DTW applied independently to each pair of seismograms in these shot records is able to recover the correct shifts.

The success of this simple method for dynamic image warping (DIW) depends on the fact that each pair of seismograms in these shot records satisfies exactly the DTW assumption that one sequence is a warped version of the other.

When this assumption is not satisfied, this simple

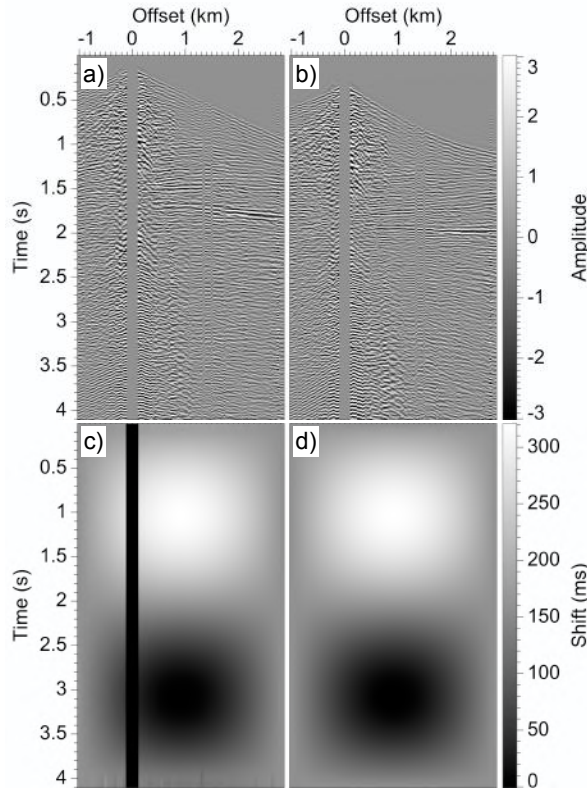


Figure 8. A recorded seismic shot record before (a) and after (b) warping with shifts that vary with time and are up to eight times larger than the dominant period of seismic reflections. Except for small offsets where data are missing, shifts (c) computed from seismograms in these two images by the dynamic time warping algorithm approximate well the known shifts (d) used to perform the actual warping.

method for DIW can fail miserably. For example, if we add different bandlimited random noise images to each of the shot records before DIW, we obtain the results shown in Figure 9. In this example, the rms signal:noise ratio is 1:1. For this noise level, it is difficult to estimate well the correct shifts from each pair of noisy seismograms in the two shot records. Therefore, the estimated shifts vary significantly for different offsets, and imply an unlimited amount of strain in the horizontal direction.

To improve these estimated shifts, we would like to limit strain in both horizontal and vertical directions. In other words, we would like to minimize alignment errors as in equations 2 and 3 while satisfying constraints like those in equations 4 or 11 in both horizontal and vertical directions.

Unfortunately, this constrained optimization problem has been shown to be NP-complete (Keyser and Unger, 2003), which means that the existence of a computationally feasible solution is highly unlikely. We must therefore make approximations to this NP-complete

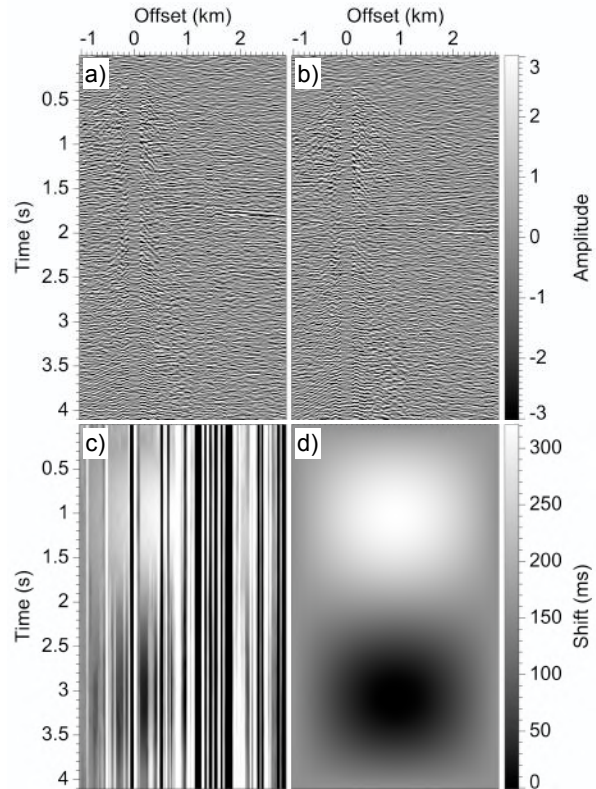


Figure 9. Same as Figure 8, except that bandlimited random noise has been added to the two shot records (a) and (b). Because these noisy seismograms are not simply related by time shifts, the shifts (c) estimated by the dynamic time warping algorithm vary wildly for different offsets, unlike the known shifts (d).

problem, and methods for DIW differ in their approximations.

4.1 Tree-sequential dynamic programming

One such approximation is that proposed by Mottl et al. (2002), and this approximation and its solution are today often referred to as *tree-sequential dynamic programming* (TSDP; e.g., Pishchulin, 2010).

Given software for DTW like that in Appendix A, implementation of the TSDP algorithm for DIW is almost trivial in the case considered here, where we seek to estimate only vertical shifts. The TSDP algorithm begins by computing alignment errors as for DTW. It then smooths those alignment errors in the vertical direction, by applying the non-linear two-sided smoothing equations 12–14 independently for each image column. TSDP ends by applying the DTW algorithm to the smoothed errors, but now in the horizontal direction, accumulating and backtracking for each image row, again independently. In this way, TSDP processes a multi-

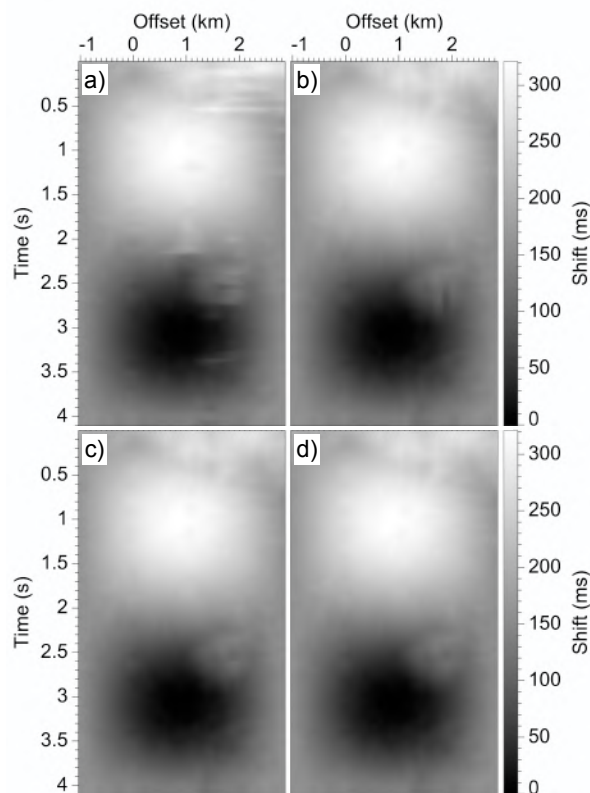


Figure 10. Time shifts estimated by dynamic time warping after vertical (a), vertical-horizontal (b), vertical-horizontal-vertical (c), and vertical-horizontal-vertical-horizontal (d) smoothings of alignment errors. Insignificant differences between shifts (c) and (d) indicate that this process of smoothing in alternating directions before DTW has converged.

dimensional image with a cascade of one-dimensional smoothing, accumulation and backtracking.

Time shifts estimated by TSDP are shown in Figure 10a. For this example I used strain limits of 25% in the vertical direction and 100% in the horizontal direction, and these values are close to the maximum strains in the known shifts displayed in Figure 9d. Compared to the estimated shifts shown in Figure 9c, the shifts from TSDP shown in Figure 10a better approximate the known shifts.

The most obvious improvement is in reduced horizontal strain, greater continuity of shifts in the horizontal direction. This improvement is not surprising, because TSDP as described above ends by applying DTW independently for each image row, and we know that DTW satisfies strain limits precisely.

However, because TSDP ends by applying DTW independently for each row, we have no guarantee that vertical strain limits (here 25%) will be satisfied. Vertical discontinuities in shifts are in fact apparent in Figure 10a. Although vertical smoothing of alignment er-

rors in TSDP reduces the likelihood that such discontinuities will occur, it does not entirely eliminate them.

Instead of smoothing vertically and then applying DTW horizontally, we might instead smooth horizontally and apply DTW vertically. Shifts estimated by this alternative implementation of TSDP are not shown here, but are significantly less accurate than those shown in Figure 10a.

The reason to first smooth vertically is that, for each offset, a pair of image columns (seismograms) typically contains multiple events that will indicate a path of minimum alignment error like that apparent in Figure 3a, but the same is not true for each pair of image rows. By first smoothing alignment errors vertically, we extend these paths of minimum error to times for which little information about vertical alignment may be available, so that DTW applied horizontally can then accurately estimate the shifts. Nevertheless, vertical discontinuities apparent in the estimated shifts shown in Figure 10a suggest that further improvement is possible.

4.2 Improving TSDP

The key to improving TSDP lies in recognizing that it first smooths alignment errors in one direction before it applies DTW in another direction. Although I have not seen TSDP described in this way, the description is accurate. So why not first smooth in both vertical and horizontal directions?

Figure 10b shows the result of smoothing both vertically and horizontally before applying DTW vertically to each column of smoothed alignment errors. As expected, vertical discontinuities in shifts are now eliminated, but a few horizontal discontinuities are apparent. However, if we apply more vertical and horizontal smoothings, this process quickly converges to the smooth shifts shown in Figure 10c and 10d.

Although I have no guarantee that this smoothing process will converge, I have not found a practical example in which more than four (vertical-horizontal-vertical-horizontal) smoothings yielded any significant changes in shifts. The convergence shown in Figure 10 is typical.

Even assuming that the smoothing process does converge, we cannot guarantee that estimated shifts will minimize alignment errors while satisfying both vertical and horizontal strain limits, as we recall that this constrained optimization problem is NP-complete (Keyser and Unger, 2003).

The new dynamic image warping method proposed here is truly an extension of the TSDP method proposed by Mottl et al. (2002). Indeed, one way to view this new method is that it is TSDP with a larger tree, in which each vertical or horizontal smoothing before dynamic time warping represents a new set of branches.

4.3 Dynamic warping and crosscorrelation

In tests of dynamic image warping discussed above, shifts are large (much larger than the dominant period of reflections) and vary rapidly with both time and offset. Recalling that strain is the rate at which shift changes, the maximum strain in time is about 25%, and the maximum strain in offset is almost 100%. That is, time shifts change by as much as a quarter of one time sample from one sampled time to the next, and by almost one time sample from one sampled offset to the next.

Where shifts are not so rapidly varying, methods based on local crosscorrelation of images may be used instead to obtain accurate shift estimates.

Figure 11 displays shifts estimated from noisy shot records like those in Figure 9a and 9b, using both dynamic image warping and local crosscorrelations. The local crosscorrelation method used here is that described by Hale (2009), which finds shifts that maximize correlation coefficients computed for seamlessly overlapping windows of images. In these tests those windows are Gaussian with half-widths equal to 320 ms in time and 240 m in offset.

Figure 11 illustrates how the success of this crosscorrelation method depends on whether or not shifts vary rapidly within the windows used to compute the correlation coefficients. The sinusoidal pattern of variation used for these tests is the same as that shown in Figure 8d, but the rates at which shifts change with time and offset (the strains) are smaller because the magnitudes of the shifts are smaller.

Where shifts vary slowly, as in Figures 11a and 11b (where the maximum time shift is less than four samples), both dynamic image warping and local crosscorrelation yield estimated shifts that approximate well the known shifts. Shifts estimated using the crosscorrelation method show significant errors only for small times and large offsets where no reflections exist.

Where shifts vary rapidly, as in Figures 11c and 11d, (ten times more rapidly than for Figures 11a and 11b), shifts estimated using the local crosscorrelation method are unstable and inaccurate, while those obtained by dynamic image warping again approximate well the known shifts.

One way to stabilize shifts estimated in crosscorrelation methods is maximize a weighted sum of both image correlation and shift smoothness. Hall (2006), for example, used such a regularization to improve the stability and accuracy of small shifts estimated from time-lapse seismic images.

Regularization is however not the same as constraints in DTW, and it does not solve the fundamental problem in using crosscorrelation windows to estimate shifts that vary rapidly within those windows. For such shifts, correlation coefficients will be low for all lags, because the two windowed images cannot be well aligned for any one of them. If we try to solve this problem

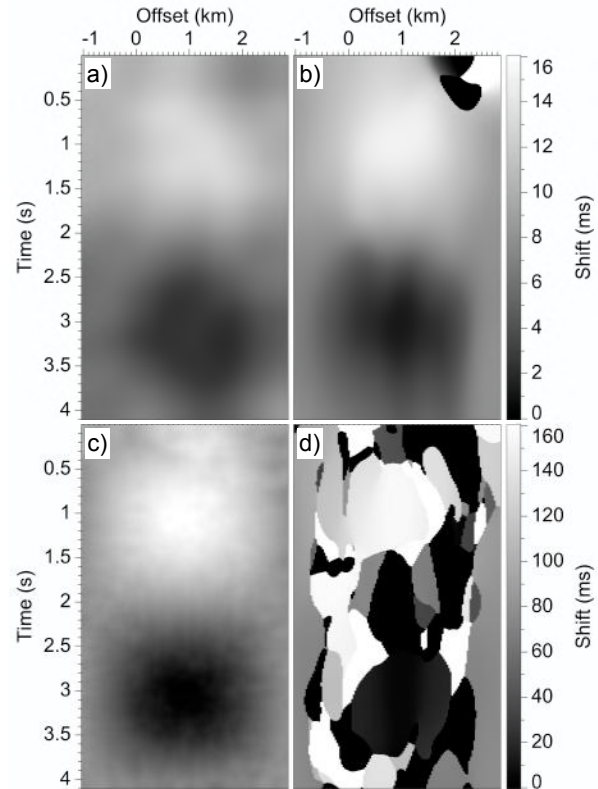


Figure 11. Time shifts estimated from noisy shot records by dynamic image warping (a,c) and by local crosscorrelation (b,d). Local crosscorrelation without constraints yields reasonable estimates where shifts vary slowly (b), but fails where shifts vary rapidly (d).

by using smaller windows, then noise will degrade our shift estimates, and in any case we must use correlation windows that are at least as large as the shifts we seek to estimate. In summary, local correlation methods require that we choose a window size; for shifts that vary rapidly, as for Figure 11d, a suitable choice may not exist.

In contrast, dynamic time and image warping require no windows. We need not specify a window size when using the dynamic image warping algorithm proposed in this paper.

Another difference between crosscorrelation and dynamic warping methods is that correlation peaks are easily found with sub-sample precision, but dynamic warping yields only integer shifts. For this reason I smooth the integer shifts estimated with dynamic warping using Gaussian filters with half-widths equal to image sampling intervals divided by maximum strains used to constrain the shifts.

5 CONCLUSION

An appealing feature of dynamic time warping is that it solves exactly the constrained optimization problem of equations 2–4. Although a practical and exact solution is unlikely to exist for the corresponding optimization problem for images, examples shown above indicate that the approximate dynamic warping solution proposed in this paper can produce reasonable estimates of shifts from noisy images, even where those shifts are large and rapidly varying.

The examples shown above are for 2D images, but dynamic image warping is easy to apply to 3D images as well. For 3D images we simply smooth alignment errors along all three image dimensions before eventually using dynamic time warping to estimate the shifts. For each dimension, this smoothing and dynamic time warping is especially efficient on computers with multiple processors, as each row or column of alignment errors can be processed in parallel.

In this paper I have assumed that images can be aligned with only vertical warping. However, the dynamic image warping algorithm proposed here can also be used to estimate shift vectors with vertical and horizontal components. Estimating shift vectors would require modification of the software in Appendix A to process arrays of alignment errors computed for multiple components of lag, but these modifications are straightforward.

ACKNOWLEDGMENTS

I am grateful to Luming Liang for emphasizing to me the shortcomings of crosscorrelation methods in estimating shifts that vary rapidly. The seismic shot record used to demonstrate dynamic image warping is one of forty shot records gathered and made freely available by Oz Yilmaz.

REFERENCES

- Anderson, K., and J. Gaby, 1983, Dynamic waveform matching: *Information Sciences*, **31**, 221–242.
- Cormen, T., C. Leiserson, R. Rivest, and C. Stein, 2001, *Introduction to algorithms*: The MIT Press.
- Hale, D., 2009, A method for estimating apparent displacement vectors from time-lapse seismic images: *Geophysics*, **74**, P99–P107.
- Hall, S., 2006, A methodology for 7d warping and deformation monitoring using time-lapse seismic data: *Geophysics*, **71**, O21–O31.
- Keyzers, D., and W. Unger, 2003, Elastic image matching is NP-complete: *Pattern Recognition Letters*, **24**, 445–453.
- Liner, C., and R. Clapp, 2004, Nonlinear pairwise alignment of seismic traces: *Geophysics*, **69**, 1552–1559.
- Mottl, V., A. Kopylov, A. Kostin, A. Yermakov, and J. Kittler, 2002, Elastic transformation of the image pixel grid for similarity based face identification: *Proceedings of the 16th International Conference on Pattern Recognition*, IEEE, 549–552.
- Müller, M., 2007, *Information retrieval for music and motion*: Springer.
- Pishchulin, L., 2010, *Matching algorithms for image recognition*: Master’s thesis, Rheinisch-Westfälischen Technischen Hochschule Aachen.
- Sakoe, H., and S. Chiba, 1978, Dynamic programming algorithm optimization for spoken word recognition: *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **26**, 43–49.

APPENDIX A: SOFTWARE

Listed below are Java methods for computing an array of accumulated alignment errors $d[i, l]$ and for backtracking within such an array to find shifts $u[0 : N - 1]$ that minimize accumulated errors. The accumulate method can be used in both positive and negative directions to compute smoothed alignment errors $\tilde{e}[i, l]$ in dynamic image warping.

Both methods can be translated easily to equivalent functions written in the C or C++ programming languages.

```

/**
 * Non-linear accumulation of alignment errors.
 * @param dir accumulation direction, + or -.
 * @param b strain is bounded by 1/b.
 * @param e input array of alignment errors.
 * @param d output array of accumulated errors.
 */
void accumulate(int dir, int b,
               float[][] e, float[][] d)
{
    int nl = e[0].length;
    int ni = e.length;
    int nlm1 = nl-1;
    int nim1 = ni-1;
    int ib = (dir>0)?0:nim1;
    int ie = (dir>0)?ni:-1;
    int is = (dir>0)?1:-1;
    for (int i=ib; i!=ie; i+=is) {
        int ji = max(0,min(nim1,i-is));
        int jb = max(0,min(nim1,i-is*b));
        for (int l=0; l<nl; ++l) {
            int lm1 = l-1; if (lm1<=-1) lm1 = 0;
            int lp1 = l+1; if (lp1==nl) lp1 = nlm1;
            float dm = d[jb][lm1];
            float di = d[ji][l ];
            float dp = d[jb][lp1];
            for (int kb=ji; kb!=jb; kb+=is) {
                dm += e[kb][lm1];
                dp += e[kb][lp1];
            }
            d[i][l] = min3(dm,di,dp)+e[i][l];
        }
    }
}

/**
 * Returns the minimum of three values.
 * If equal values, choose the value b.
 */
float min3(float a, float b, float c) {
    return b<=a?(b<=c?b:c):(a<=c?a:c);
}

```

```

/**
 * Finds shifts by backtracking within an
 * array of accumulated alignment errors.
 * Backtracking must be performed in the
 * direction opposite that in which the
 * alignment errors were accumulated.
 * @param dir backtrack direction, + or -.
 * @param b strain is bounded by 1/b.
 * @param lmin lag for lag index 0.
 * @param d input array of accumulated errors.
 * @param e input array of alignment errors.
 * @param u output array of shifts.
 */
void backtrack(int dir, int b, int lmin,
               float[][] d, float[][] e, float[] u)
{
    int nl = d[0].length;
    int ni = d.length;
    int nlm1 = nl-1;
    int nim1 = ni-1;
    int ib = (dir>0)?0:nim1;
    int ie = (dir>0)?nim1:0;
    int is = (dir>0)?1:-1;
    int ii = ib;
    int il = 0;
    float dl = d[ii][il];
    for (int jl=1; jl<nl; ++jl) {
        if (d[ii][jl]<dl) {
            dl = d[ii][jl];
            il = jl;
        }
    }
    u[ii] = il+lmin;
    while (ii!=ie) {
        int ji = max(0,min(nim1,ii+is));
        int jb = max(0,min(nim1,ii+is*b));
        int ilm1 = il-1; if (ilm1<=-1) ilm1 = 0;
        int ilp1 = il+1; if (ilp1==nl) ilp1 = nlm1;
        float dm = d[jb][ilm1];
        float di = d[ji][il ];
        float dp = d[jb][ilp1];
        for (int kb=ji; kb!=jb; kb+=is) {
            dm += e[kb][ilm1];
            dp += e[kb][ilp1];
        }
        dl = min3(dm,di,dp);
        if (dl!=di) {
            if (dl==dm) {
                il = ilm1;
            } else {
                il = ilp1;
            }
        }
    }
    ii += is;
    u[ii] = il+lmin;
    if (il==ilm1 || il==ilp1) {
        for (int kb=ji; kb!=jb; kb+=is) {
            ii += is;
            u[ii] = il+lmin;
        }
    }
}
}

```

