Stanford Exploration Project, Report 70, December 18, 1997, pages 1–300

Stanford Exploration Project, Report 70, December 18, 1997, pages 1–300

Automatic picking and its applications

Lin Zhang[1]

keywords: *algorithm, velocity, modeling, optimization, interpolation*

ABSTRACT

I have developed a new automatic picking algorithm that can be applied to many picking problems in seismic data processing. This algorithm does the initial picking by solving a constrained non-linear optimization problem with a fast search algorithm. Then, by linearizing the model of data, the objective function defined in the optimization process is approximately reduced to a quadratic form. The residual corrections are obtained by solving a linear equation.
I applied this algorithm to several practical problems, including dip-picking, event-picking, well-log interpolation and velocity picking. Examples with field data show that the results are reasonably accurate and reliable.

INTRODUCTION

Picking is a process of identifying the events or estimating the parameters of the events on seismic sections. This process can be done manually, or automatically by computers. Because human eyes have a strong capability to recognize seismic events on a noisy background, manual picking is often more robust than any computer algorithm. However, manual picking has the disadvantage of being inaccurate in estimating the parameters of events, and is inefficient and expensive. When a seismic section contains many events, automatic techniques should be incorporated in the picking process.

Automatic picking is usually accomplished through optimization. It involves two steps: defining an objective function and optimization picking. Two types of objective functions commonly used are error measures (Martinson et al., 1982; Zhang and Claerbout, 1990) and coherence measures (Leaney and Ulrych, 1987; Biondi, 1990). Optimal picking either minimizes an error measure or maximizes a coherence measure. A variety of algorithms have been employed to extremize objective functions. Among them, iterative methods and search methods are the most popular ones. Iterative methods are computationally efficient, but have two drawbacks: they require an initial solution, and, for non-linear optimization problems, they may not find the globally

[1]**email:** not available

optimal solution. Search methods do not have these two problems, however, they are often computationally expensive, and sometimes even impossible to implement. The accuracy of a serach method depends on the increment at which the solution space is sampled.

In this paper, I generalize the concept of error measure from two channels to multichannel, and define picking as a process of estimating parameters through constrained optimization. I show that if an L_2 norm is used, minimizing an error measure is equivalent to maximizing a coherence measure. I use a fast search algorithm to get an initial solution to the non-linear optimization problem, and then refine the solution by estimating the residuals through linear optimization. I apply this method to several picking problems in data processing and obtain satisfactory results.

The first section of this paper describes different objective functions and their relations. The following sections explain how to solve the constrained non-linear optimization problem to obtain an initial pick and how to estimate residual corrections to improve the results. The fourth section shows the application of the method to specific problems, such as dip-picking, event-picking, well-log interpolation and velocity picking. Finally, I will discuss possible improvements and extensions of the method.

OBJECTIVE FUNCTIONS

The formulation of an optimal estimation usually consists of two steps. The first step is to construct an objective function that depends on the unknown parameter to be determined. This function measures the quality of the estimation. The second step is to estimate the unknown parameters through the extremization of the objective function, depending on type of the quality measure. In this section, I use dip estimation of seismic events as an example to describe two types of quality measures.

Let us suppose a seismic section $P(t,x)$ contains events reflected from subsurface boundaries. We want to estimate the local dips of these events. An event with local dip p at point (t,x) follows the trajectory

$$\hat{t} = t + p(\hat{x} - x) \tag{1}$$

at that point. For each point on the section, we can construct a $2L_t \times 2L_x$ subsection by applying linear moveout corrections to a window of data

$$P_{ij} = P(t + i\Delta t + jp\Delta x, x + j\Delta x). \tag{2}$$

where $-L_t \leq i \leq L_t$ and $-L_x \leq j \leq L_x$. The symbols Δt and Δx are the temporal and spatial sampling intervals, respectively. If the local dip p used for the linear moveout correction is equal to the true dip at the location (t,x), then P_{ij} is composed of horizontal events. Otherwise, the events in P_{ij} are slanted. With this observation, we can define the objective functions.

Error measure

The error in estimating p can not be directly evaluated because the true value of p is unknown. However, a function of p that is related to this error can be evaluated:

$$E(t,x,p) = \sum_{i=-L_t}^{L_t} W_t(i) \sum_{j,k=-L_x}^{L_x} W_x(j)W_x(k)(P_{ij}-P_{ik})^2, \tag{3}$$

where W_t and W_x are weighting functions. As a function of p, $E(t,x,p)$ measures the differences between the traces within the subsection P_{ij}. This function reaches its minimum value when the estimated p is equal to the true dip at (t,x). Therefore, it can be used as an error measure.

It may be desirable to have an error measure that is bounded between 0 and 1. This can be easily done through normalization:

$$E_n(t,x,p) = \sum_{i=-L_t}^{L_t} W_t(i) \frac{\sum_{j,k=-L_x}^{L_x} W_x(j)W_x(k)(P_{ij}-P_{ik})^2}{\sum_{j=-L_x}^{L_x} W_x(j)P_{ij}^2}. \tag{4}$$

One can estimate p by minimizing either $E(x,t,p)$ or $E_n(x,t,p)$. If the data contain rich low-frequency components, such as a smoothed well log, equation (3) should be used. For seismic data, the normalized error measure should be used.

Coherence measure

Another kind of quality measure for an estimation process is a coherence measure:

$$C(t,x,p) = \sum_{i=-L_t}^{L_t} W_t(i) \left[\sum_{j=-L_x}^{L_x} W_x(j)P_{ij}\right]^2. \tag{5}$$

Coherence measures have been previously used in seismic data processing, for example, in velocity analysis (Biondi, 1990). Its normalized version is called semblance:

$$C_n(t,x,p) = \sum_{i=-L_t}^{L_t} W_t(i) \frac{\left[\sum_{j=-L_x}^{L_x} W_x(j)P_{ij}\right]^2}{\sum_{j=-L_x}^{L_x} W_x(j)P_{ij}^2}. \tag{6}$$

It can be shown that both $C(t,x,p)$ and $C_n(t,x,p)$ reach their maximum values when the traces in the subsection $P_{i,j}$ are identical, which implies that the estimated p is equal to the true dip.

Relations

We can achieve the optimal estimation by minimizing the functions in equations (3) and (4), or maximizing the functions in equations (5) and (6). Would we get an identical result if we use the two different methods? To answer this question, let us examine the relations between these objective functions:

$$E(t,x,p) = \sum_{i=-L_t}^{L_t} W_t(i) \sum_{j=-L_x}^{L_x} W_x(j) P_{ij}^2 - C(t,x,p), \tag{7}$$

and

$$E_n(t,x,p) = 1 - C_n(t,x,p). \tag{8}$$

It is clear that, for the normalized objective functions, minimizing $E_n(t,x,p)$ is completely equivalent to maximizing $C_n(t,x,p)$. The first term on the right-hand side of equation (7) is the weighted energy of the subsection. This term usually maintains to be constant. Therefore, for unnormalized objective functions, we can draw a similar conclusion to that of the normalized objective functions. In view of the calculation of these functions, the coherence measures are easier to calculate than the error measures.

NON-LINEAR OPTIMIZATION

In the last section, I showed that automatic dip-picking can be accomplished by solving an optimization problem. Because the objection functions I considered are non-quadratic functions of unknown parameter p, the optimization problem is non-linear. In this section, I use a fast search algorithm to solve this non-linear optimization.

Unconstrained optimization

A straightforward algorithm for estimating p at (t,x) is to scan the objective function $E_n(t,x,p)$ over a reasonable range of p, and then find the minimizer of the objective function with a 1-D search. The dip obtained with this algorithm is an unconstrained solution to the problem because p can be any arbitrary function of (t,x). This unconstrained solution is sensitive to noise in the data; it may be completely wrong when the data are insufficiently sampled.

Constrained optimization

Incorporating a predetermined model in the searching procedure usually produces a robust algorithm. The model constrains p to be certain type of functions of (t,x), for example, smoothed functions. This constraint excludes many possible estimation errors caused by noise and aliasing. Of course, the choice of predetermined model depends on the individual application. Ideally, the function should be constrained in both axes, however, doing this significantly complicates the search algorithm. Thus, I decide to constrain the solution in t axis only, which proves to be practical for many applications.

The constrained optimization can be formally stated as follows: for each x, find a function $p(t)$ that satisfies certain constraints and minimizes

$$Q = \sum_{p(t)} E_n(t, x, p).$$

Because any function $p(t)$ defines a path through the t-p plane, an alternative description of the constrained optimization is to find a constrained path $p(t)$ in the t-p plane such that the total error measure summed along this path is minimum. Clearly, this is a shortest-path problem. There is an efficient and systematic algorithm for solving this type of problem.

Viterbi algorithm

The Viterbi algorithm is an optimized searching algorithm for finding the shortest path. It has been used to decode the convolutional codes for telecommunication. The key principle of this algorithm arises from the observation that if the shortest path between point A and point C passes a intermediate point B, then the segment of this path between point A and point B is the shortest path between these two points. The algorithm consists of two steps: forward shortest-path accumulation and backward descending tracking. One can find the details of the algorithm in an excellent reference (McEliece, 1977). I include a subroutine of the Viterbi algorithm in Appendix A.

If the coherence measure in equations (5) or (6) is chosen as the objective function, then the algorithm should search for the longest-path, instead of shortest path. This can be easily done by reversing the operations in the Viterbi algorithm.

At this point, I have not mentioned anything about the constraints imposed by a model. In fact, simple models can be easily incorporated in the Viterbi algorithm. For example, to obtain a smooth solution $p(t)$, one can eliminate all the transition branches that would cause abrupt changes of the path. This procedure provides two benefits; not only does it constrain the smoothness of the solution, but it also reduces the computation time because the solution space to be searched is reduced. Field data examples show that this procedure works well for many applications.

LINEAR OPTIMIZATION

Search algorithms usually have a common pitfall. The accuracy of its solution is limited to the increment at which the solution space is sampled, as is the algorithm described in the last section. One way to solve this problem is to reduce the sampling interval. However, doing this increases the size of the solution space, which, in turn, increases the computational cost substantially. I avoid this problem by estimating residual dips through a linear optimization.

Residual dip

Let us suppose p_0 is a dip picked at a point through the constrained non-linear optimization and p is the true dip at that point. The difference between them is a small residual dip Δp.

$$p = p_0 + \Delta p \tag{9}$$

We can substitute p in equation (2) with equation (9) and linearize the expression on the left-hand side of equation (2):

$$\begin{aligned} P_{ij} &= P(t + i\Delta t + jp\Delta x, x + j\Delta x) \\ &\approx P(t + i\Delta t + jp_0\Delta x, x + j\Delta x) + \frac{\partial P}{\partial t} j\Delta p\Delta x \\ &= P_{ij}^0 + \frac{\partial P}{\partial t} j\Delta p\Delta x, \end{aligned} \tag{10}$$

to obtain P_{ij} as a linear function of the residual dip Δp.

Objective function and solution

We can apply the approximation in equation (10) to the objective functions. For example, from equations (3) and (7) we can derive

$$\begin{aligned} E^0(t, x, \Delta p) = \sum_{i=-L_t}^{L_t} W_t(i) \Bigg\{ & \sum_{j=-L_x}^{L_x} W_x(j) \left(P_{ij}^0 + \frac{\partial P}{\partial t} j\Delta p\Delta x \right)^2 \\ & - \left[\sum_{j=-L_x}^{L_x} W_x(j) \left(P_{ij}^0 + \frac{\partial P}{\partial t} j\Delta p\Delta x \right) \right]^2 \Bigg\} \end{aligned} \tag{11}$$

The optimal estimate of Δp is the minimizer of this function. Because the objective function is a quadratic function of the unknown Δp, one can use the standard least-squares techniques to find the solution of this linear optimization problem:

$$\Delta p = \frac{\sum_{i=-L_t}^{L_t} W_t(i) \left[\sum_{j=-L_x}^{L_x} W_x(j) P_{ij}^0 \sum_{j=-L_x}^{L_x} W_x(j) \frac{\partial P}{\partial t} j\Delta x - \sum_{j=-L_x}^{L_x} W_x(j) P_{ij}^0 \frac{\partial P}{\partial t} j\Delta x \right]}{\sum_{i=-L_t}^{L_t} W_t(i) \left[\sum_{j=-L_x}^{L_x} W_x(j) \left(\frac{\partial P}{\partial t} j\Delta x \right)^2 - \left(\sum_{j=-L_x}^{L_x} W_x(j) \frac{\partial P}{\partial t} j\Delta x \right)^2 \right]} \tag{12}$$

Once Δp is found for each t, the pick can be improved by using equation (9).

APPLICATIONS

In this section, I show the applications of the automatic-picking algorithm to dip-picking, event-picking, well-log interpolation and velocity picking.

Dip-picking and event-picking

Many algorithms in seismic data processing assume knowledge of the local dips of events on seismic sections. These local dips can be picked automatically by using the algorithm developed in the previous sections. Once the local dips are known on a uniform grid, we can trace the events on a seismic section. This process is called event-picking and has applications in structural interpretations and velocity analysis (Cunha Filho, 1991).

To pick an event, I first manually identify the event at one location, and then let the event-picking program trace the event to both sides using the equation:

$$t_{j\pm 1} = t_j \pm p_j \Delta x + \frac{1}{2}\frac{dp}{dx}(\Delta x)^2. \tag{13}$$

This analogous to the problem of finding the stream lines of a field when the gradients of the field are known. One may find a better method of solution, nevertheless, field data examples show that my method is adequate for these seismic applications.

I applied my automatic picking algorithm to two field data gathers. The first one is a common midpoint (CMP) gather on which the events are seriously aliased at far offsets. The results of dip-picking are shown in Figure **??**a. The picked local dips are displayed by the small line segments whose dips are equal to the dips of the events at those locations. We see that the small line segments are always parallel to the events shown on the background. The picked events are plotted on top of the raster image of the CMP gather, as shown in Figure **??**b. The picked curves follow the correct trajectories of the events and ignore the coherent noises caused by aliasing.

To understand why this picking algorithm has an anti-aliasing property, let us look at an example of the objective function computed in the step of the constrained non-linear optimization. Figure **??** shows the objective function $E_n(t, x, p)$ for a fixed x. We see a sequence of minima that form a path from the left to the right. We also see some local minima that do not form complete paths. These local minima are created by the aliasing noise. If the unconstrained optimization is used, the algorithm might mistakenly pick some of the local minima. The constrained optimization forces the algorithm to pick a complete shortest-path that is displayed by the solid curve in Figure **??**. For the same reason, the algorithm is not too insensitive to noise in data.

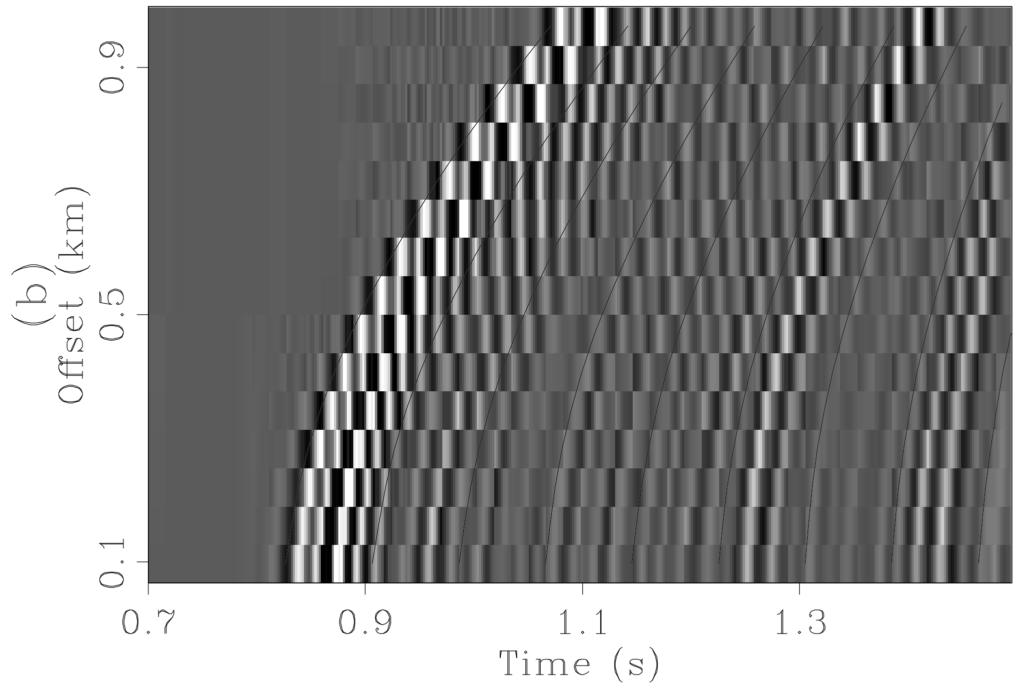
The second field data gather is a common-receiver gather recorded in a walk-away marine survey. This gather is collected with many shots on the surface and a receiver down in a well. The first arrival and several events that follow it are down-going waves. After 0.75 seconds, the up-going waves reflected from a deep layer start to interfere with the down-going waves. The picked local dips are represented by small line segments plotted in Figure **??**a, and the picked events are displayed by the solid curves shown in Figure **??**. Clearly, the algorithm performs reasonably well. It has no problem to pick the dip at locations where a single event is present. At locations where two or more events interfere with each other, the algorithm picks the dip of the event that is relatively stronger and whose dip value does not create an abrupt change of the dip function $p(t)$. Looking carefully at the picked curve for the first arrival, we see that a "pull-down" of the first arrival at offset -0.6 km is correctly picked. This "pull-down" may be caused by a low velocity anomaly in the medium. This observation suggests a good accuracy of the algorithm. This event-picking method will probably be applied to traveltime inversion.

Well-log interpolation

One of the commonly used methods for velocity inversion is the iterative method. To start the inversion process, one needs to have an initial velocity model. The updated velocity model is obtained by perturbing the previous velocity model in a way. To control the stability of the inversion process, one can set up a constraint to penalize large perturbations in the the new velocity model from a predetermined model. Usually a constant velocity model or a smooth velocity model extracted from data is used for both the initial model and the predetermined model. Actually, it would be nice that these models are constructed from some independent and hard data, such as sonic log measurements from wells that are located along the seismic survey lines.

To create a 2-D velocity model from sonic logs, one needs to do interpolation. Wells are often drilled with large separating distances. Data collected with such a large sample interval are often believed to be seriously aliased. Is it possible to interpolate this kind of data? The answer is positive if we are interested in having a smooth 2-D velocity model. Because smooth sonic logs are non-Gaussian, they can be interpolated with a non-linear method.

The next question one may ask is how the interpolation should be done. My idea is that this kind of data should be interpolated along its contour lines because data samples along these trajectories have constant values. Based on this idea, I developed a two-step algorithm to interpolate this kind of data. First, I use the known well logs to predict the contour lines of data. Then I interpolate data along these contour lines. Predicting contour lines for such sparsely sampled data is an underdetermined problem. I use a model to parameterize the contour lines. If only two well logs are available, the contour lines are straight lines. If three or more well logs are available, then the contour lines are modeled by cubic-spline functions. These models are parameterized by the depth positions of the contour lines at the locations of the known wells. Once we find these depth positions, we can generate the whole set of contour lines.



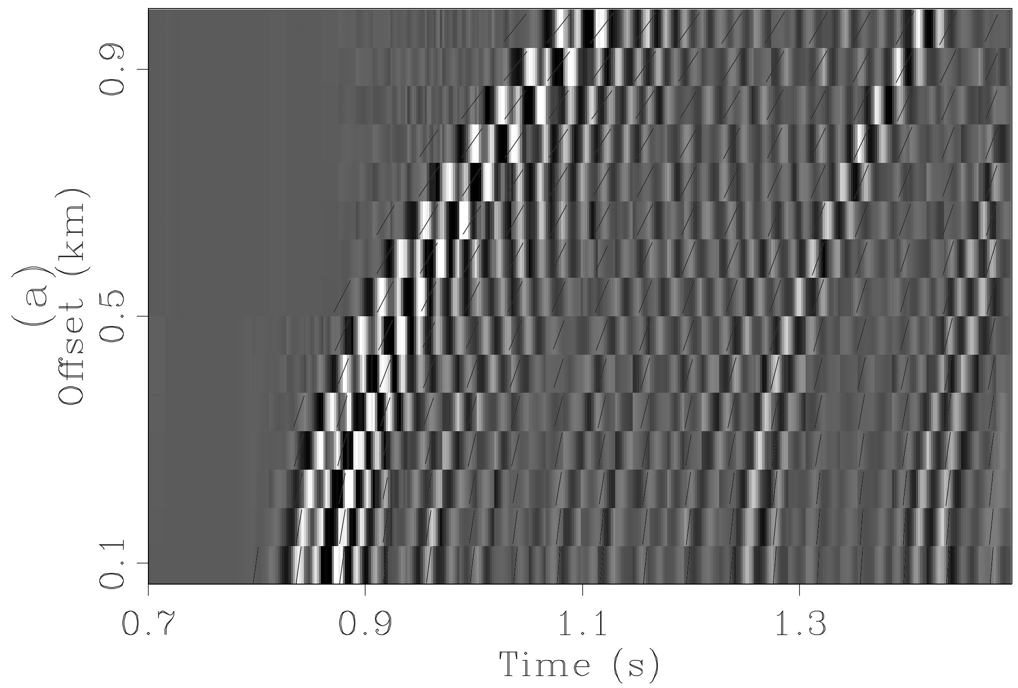


Figure 1: Picking of a CMP gather: (a) dip-picking; (b) event-picking. lin2-cmppick [NR]

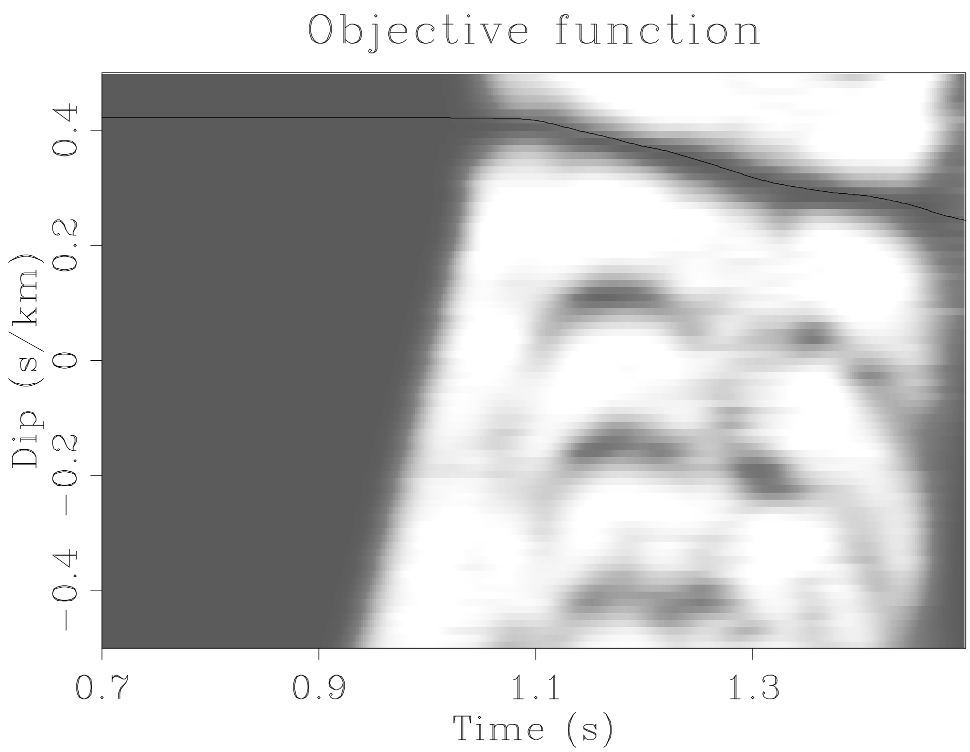


Figure 2: Objective function for a fixed x. The solid curve shows the shortest path. High intensities are large values. lin2-cmpobj [NR]

Suppose a contour has a depth position z at well 1 and a depth position $z + \Delta z$ at well 2. Then the log sample $u_1(z)$ at well 1 should be equal to the log sample $u_2(z + \Delta z)$ because they are on the same contour line. Therefore, we can estimate Δz by minimizing the difference between $u_1(z)$ and $u_2(z + \Delta z)$. Obviously, Δz is a function of z. We can estimate the function $\Delta z(z)$ through the constrained non-linear optimization as we did for dip estimation. The constraint for this problem is that the contour lines do not cross each other. When three or more well logs are available, $\Delta z(z)$ is estimated for each pair of well logs. From these functions, I generate all contour lines and interpolate data along these contour lines.

I used a synthetic model to test my interpolation algorithm. The top panel of Figure **??** shows a synthetic slowness model. The model has fairly complicated subsurface structure. The interval slowness within each layer is constant. The middle panel of the figure shows seven slowness profiles that simulates the well logs measured from the synthetic model and with a large separating distance. The bottom panel of Figure **??** displays the slowness model interpolated from those slowness profiles. The interfaces of the original model are plotted on the same panel. The algorithm does a very good job. The interpolated model is very close to the original model except at the pinch-out.

Now I show some field data examples. Figure **??**a displays three smoothed sonic

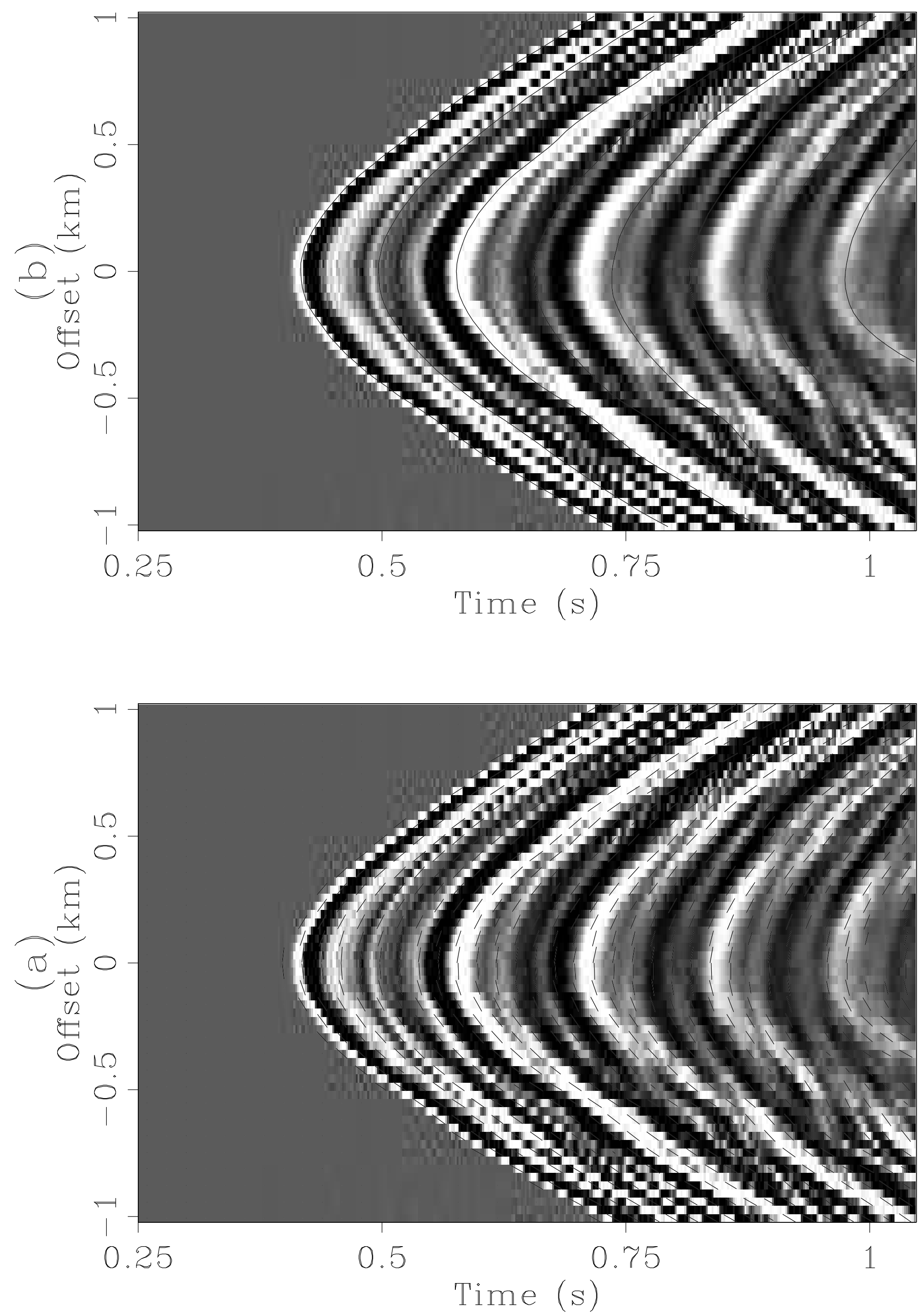


Figure 3: Picking of a common receiver gather: (a) dip-picking; (b) event-picking.
lin2-vsppick [NR]

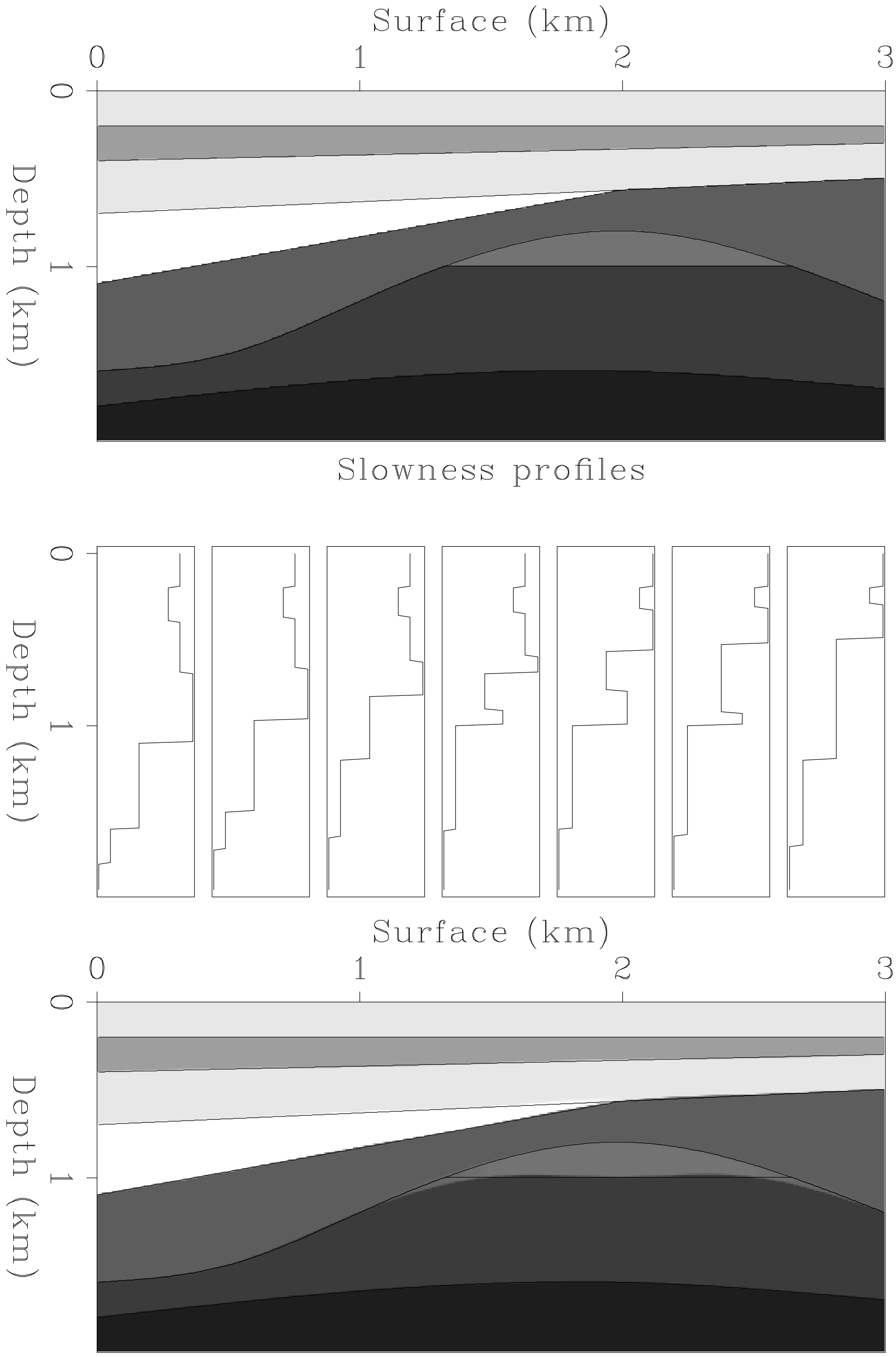


Figure 4: A synthetic example. Top: a slowness model. Middle: 7 well logs. Bottom: the interpolated slowness model. lin2-slowint [NR]

logs. The distances between wells are several kilometers. Figure **??**b shows the result of interpolation. The interpolated model is smooth. It has a well-defined high velocity layer at depth of 3 km and a well-defined low velocity layer at the depth of 4 km. Unfortunately, I am not able to check the accuracy of this result.

The well logs shown in Figure **??** are collected from two wells between which a cross-well experiment was conducted. The middle panel of this figure displays the interpolated slowness model. The dashed lines on this panel are the geological structures interpreted from an independent source (Harris et al., 1990). It is clear that the two results agree fairly well.

Velocity picking

Conventional velocity analysis is done in two steps. The first step is to compute the semblances from CMP gathers. The second step is to pick a velocity model that maximizes the semblances. It is clear that if equation (1) is replaced by the Dix equation

$$\hat{t}^2 = t^2 + \frac{\hat{x}^2}{v_{RMS}^2}, \tag{14}$$

then the algorithm described earlier can be used for velocity analysis. The constraint required in this problem is a smooth RMS velocity function. Figure **??**a shows a CMP gather from the Gulf of Mexico. The semblance computed from this gather is displayed in Figure **??**b. I overlay the semblance with the picked velocity functions. The stars indicates the picks when unconstrained optimization is used. These picks form a rough and unrealistic velocity function. By using a weak smoothness constraint, I get a velocity function plotted with the dashed curve. Now the result becomes more acceptable. But still, the algorithm picks the peak around 2.3 second, that is known to be caused by a dipping bed. I further increase the smoothness constraint and obtain the final picks plotted with a solid curve in Figure **??**b. This result is reasonably good.

CONCLUSIONS

I have described an automatic picking algorithm that combines linear and non-linear optimization procedures. The algorithm is applied to dip-picking, event-picking, well-log interpolation and velocity picking. The results are reasonably accurate and reliable. For specific applications, the linear optimization part can be further improved. For example, in velocity picking, one can use Toldi's iterative method (Toldi, 1985). The event-picking may have an application in the traveltime picking step of tomographic inversion. I intend to incorporate the local dip information into imaging operators so that DMO and migration become a one-to-one mapping. An important extension of the algorithm is to allow multiple dips. The success of this extension will make the algorithm much more valuable than it is now.

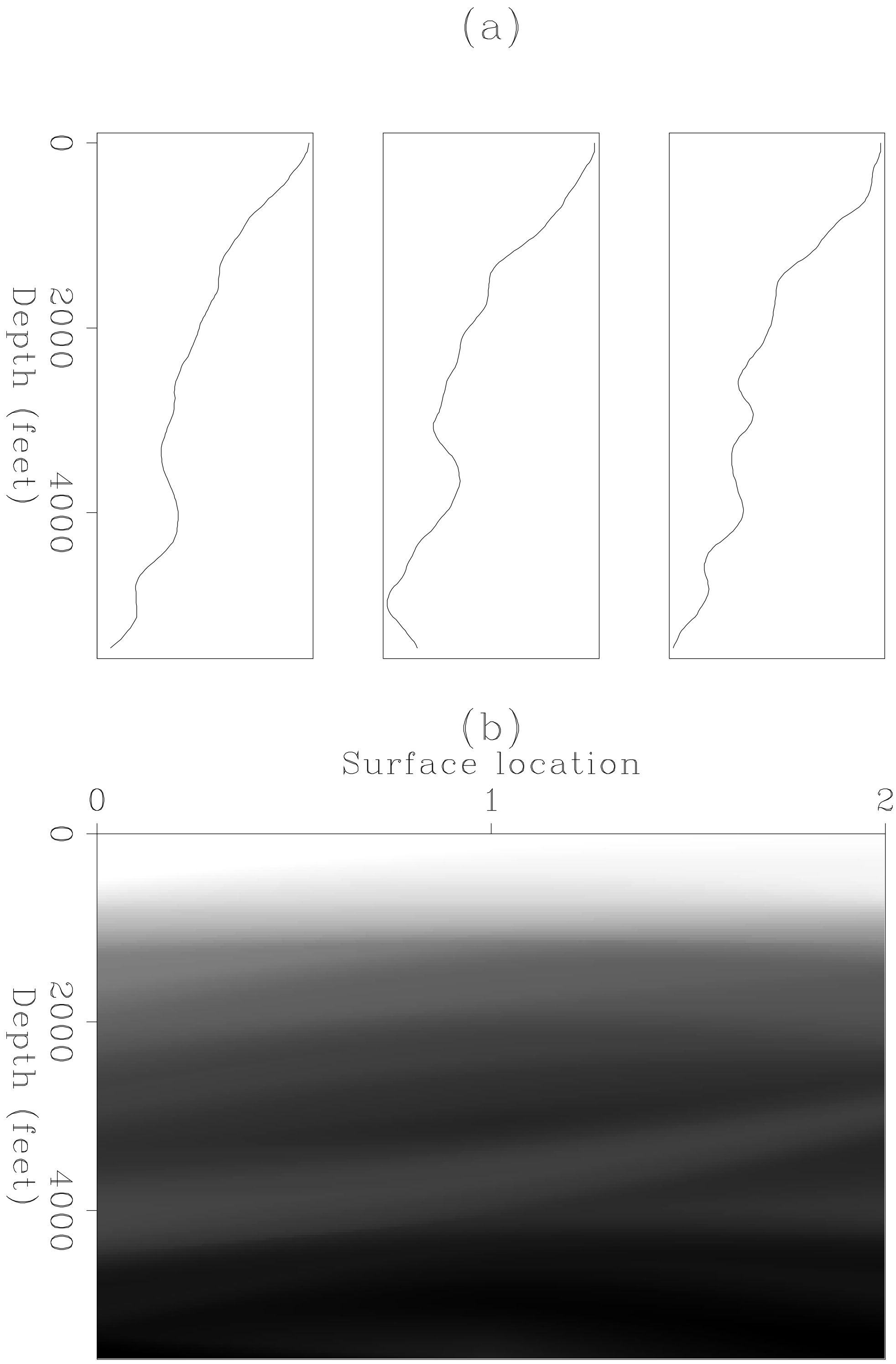


Figure 5: Interpolation of sonic logs: (a) two sonic logs; (b) interpolated slowness model $\boxed{\text{lin2-sonicint}}$ [NR]

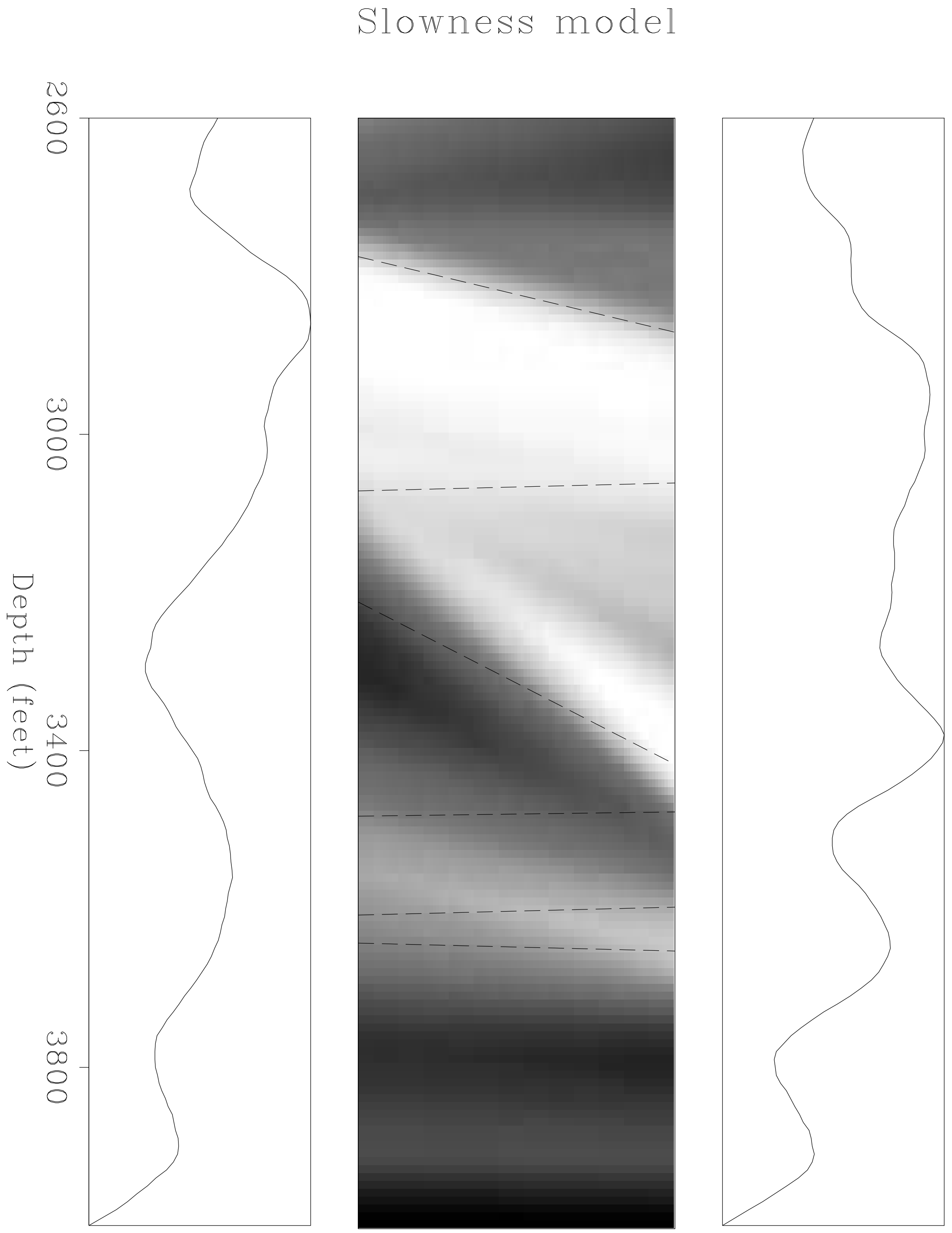


Figure 6: Interpolation of a sonic log. Left: well log A. Right: well log B. Middle: interpolated slowness model. lin2-tomoint [NR]

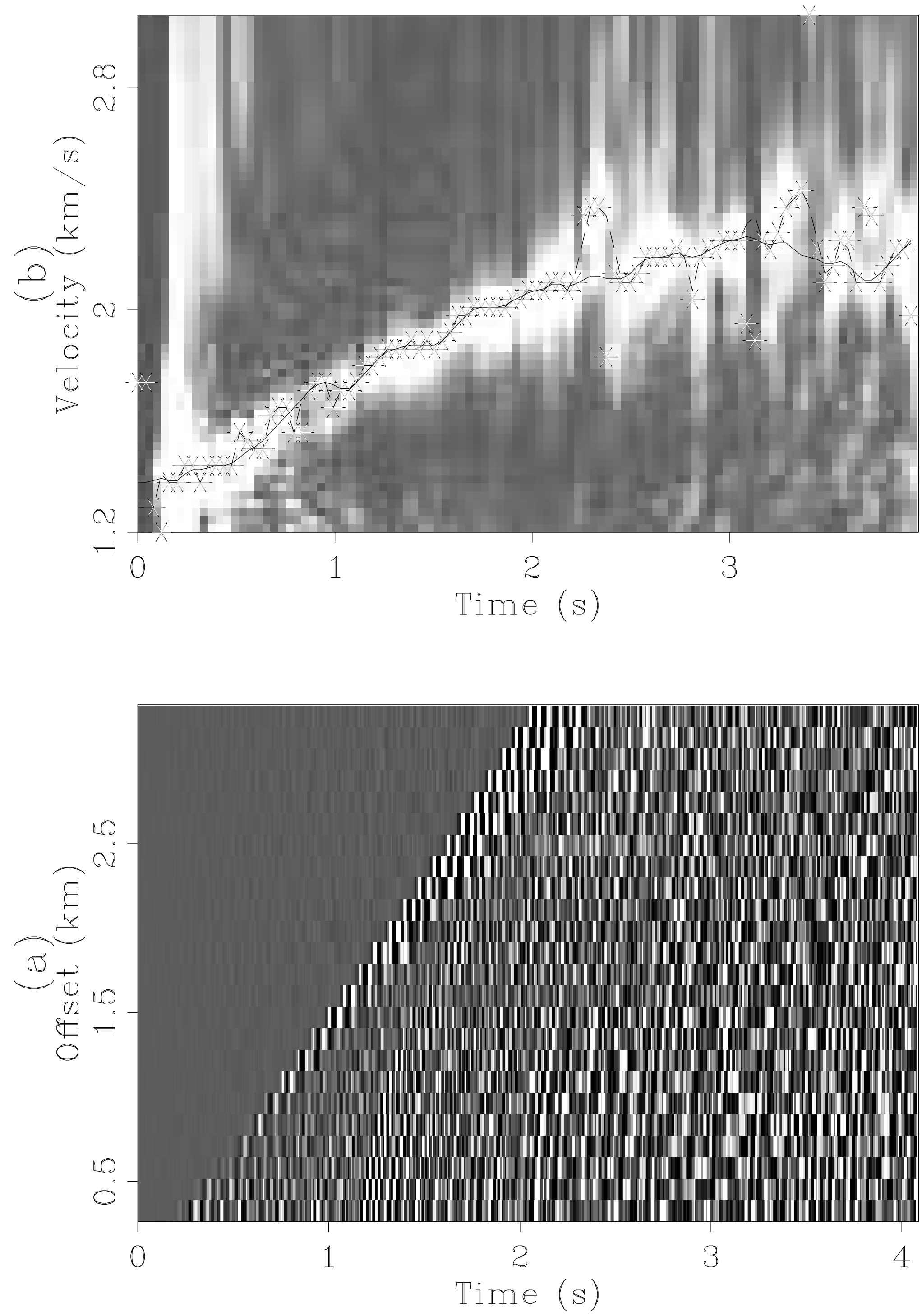


Figure 7: Velocity picking: (a) a common midpoint gather; (b) semblance overlain with the results of velocity picking. The stars show the locations of the maximum of semblance for each time samples. The dashed curve shows the velocity picking with a weak constraint. The final result is shown by the solid curve. lin2-halevel [NR]

ACKNOWLEDGMENTS

I thank Stanford tomography project for providing the well logs used in this paper.

REFERENCES

Biondi, B., 1990, Seismic velocity estimation by beam stack: Ph.D. Thesis, Stanford University.

Cunha Filho, C., 1991, Interval velocity estimation with event picking: SEP–**70**.

Harris, J.M., Tan, H., Lines, L., Pearson, C., and Treitel, S., 1990, Cross-well tomographic imaging of geological structures in gulf coast sediments: Presented at the 60th Annual International SEG meeting in San Francisco.

Leaney, S.W., and Ulrych T.J., 1987, Multiple dynamic matching: a new approach to well log correlation: Geoexploration, **24**, 503-515.

Martinson, D. G., Menke, W., and Stoffa, P., 1982, An inverse approach to signal correlation: J. Geophys. Res., **87**, 4807-4818.

McEliece, R.J., 1977, The theory of information and coding: Addison-Wesley Publishing Company.

Toldi, J., 1985, Velocity analysis without picking: Ph.D. Thesis, Stanford University.

Zhang L. and Claerbout J., 1990, Autopick dip-picking by non-linear optimization: SEP–**67**, 123-138.

APPENDIX A

The following Fortran subroutine implements the Viterbi algorithm:

```
        subroutine rviterbi(n1,n2,npath,in,out,qqsum,qqmin)
c
c  ------------------------------------------------------------------------
c               find the shortest-path with viterbi algorithm
c  ------------------------------------------------------------------------
c
c       n1,n2                           dimension of the array
c       npath                           number of allowed paths
c       in                              input array
```

```
c       qqsum,qqmin                         auxiliary array
c       out                                 output array
c
c  -----------------------------------------------------------------------
c
        implicit none

        integer n1,n2,i1,i2
        integer npath,ipath
        integer i2min,i2max,npathh,ipathmin,ipathmax
        integer i2pick,i2newpick
        real minvalue
        real in(n1,n2)
        real qqsum(n1,n2),qqmin(n2)
        real out(n1)

        npathh = (npath-1)/2
c VITERBI ALGORITHM
c    forward error accumulation
        call zero(n2,qqmin)
        call zero(n1*n2,qqsum)
        do i2 = 1,n2
           qqsum(1,i2) = in(1,i2)
        enddo
        do i1 = 2,n1
           do i2 = 1,n2
              qqmin(i2) = 1.e+30
           enddo
           do ipath = -npathh,npathh
              i2min = max(1,1-ipath)
              i2max = min(n2,n2-ipath)
              do i2 = i2min,i2max
                 qqmin(i2) = min(qqmin(i2),qqsum(i1-1,i2+ipath))
              enddo
           enddo
           do i2 = 1,n2
              qqsum(i1,i2) = qqmin(i2)+in(i1,i2)
           enddo
        enddo

c    backward tracing
        i1 = n1
        i2pick = (n2-1)/2
        minvalue = qqsum(i1,i2pick)
        do i2 = 1,n2
           if(qqsum(i1,i2) .lt. minvalue)then
              i2pick = i2
              minvalue = qqsum(i1,i2pick)
           endif
        enddo
        out(i1) = real(i2pick)
        do i1 = n1-1,1,-1
           minvalue = 1.e+30
           if(qqsum(i1,i2pick) .le. minvalue)then
```

```
      minvalue = qqsum(i1,i2pick)
      i2newpick = i2pick
   endif
   ipathmin = max(-npathh,1-i2pick)
   do ipath = ipathmin,-1
      if(qqsum(i1,i2pick+ipath) .lt. minvalue)then
         minvalue = qqsum(i1,i2pick+ipath)
         i2newpick = i2pick+ipath
      endif
   enddo
   ipathmax = min(npathh,n2-i2pick)
   do ipath = 1,ipathmax
      if(qqsum(i1,i2pick+ipath) .lt. minvalue)then
         minvalue = qqsum(i1,i2pick+ipath)
         i2newpick = i2pick+ipath
      endif
   enddo
   i2pick = i2newpick
   out(i1) = real(i2pick)
enddo

return
end
```