Lloyd and Viterbi for QC and auto-picking

Robert G. Clapp

ABSTRACT

Automatic picking and the QCing of these picks are crucial step in the velocity analysis loop. In this paper I show that a modified version of Viterbi's algorithm can be an effective auto-picker when used interactively. In addition I show that Lloyd's algorithm can reduce densely auto-picked information to a representative subset that simplifies the QCing process.

INTRODUCTION

To speed up the velocity analysis loop it is important to allow the computer to do as much as work as possible, particularly in the human intensive picking portion of the loop. As essential is an easy way to evaluate and modify (QC) these automatic picks.

Dynamic programming is an effective tool for finding a solution for certain types of relatively small, non-linear problems such as semblance picking. In biology, dynamic programming is used for pairwise alignment of amino acid sequences (Needleman and Wunsch, 1970). In electrical engineering, it is used for error correction in wireless communication and speech recognition (Hosom et al., 1999) among many other things. We can also find examples of its use in geophysics. Kruse (1988) used dynamic programming for signal correlation and trace interpolation. Kruse (1988) calculates an error function based on the difference in instantaneous frequency between all points along two signals. Dynamic programming is then used to find the error path with the least energy. Liner and Clapp (2004) used dynamic programming for alignment. Zhang (1991) used it for a starting solution when doing event picking.

Quantization is an important field in both electrical engineering and computer graphics. In speech compression, it is important to accurately describe a signal in as few bytes as possible. In image processing, it is often important to reduce the number of colors in image with as little loss in image quality as possible. One family of method often employed in quantization is based on Lloyd's method (Lloyd, 1982), an iterative technique that allows for variable rate quantization. For QCing automatic picks, the ability to reduce densely picked functions to a smaller set of representative points simplifies substantially the QCing process.

In this paper I use a dynamic programming technique to automatically pick semblance gathers and reflectors. In addition, I show that quantization can be used to represent a function by a subset of representative points.

VITERBI

Dynamic programming, and specifically the Viterbi algorithm, offers a way to solve certain classes of non-linear problems. It is useful for problems that can be thought of as making one decision after another. To understand how it works, it is easiest to start with our final decision. Our goal is to maximize our 'score' S of a series of decision $1...n$. Each decision has several potential outcomes $(1...m)$. Our final score is going to be based on some score we have calculated for all of our possible options (called 'states') at $j = n - 1$, and the best score we can get from moving from all of the possible states at $n - 1$ to all possible states at n. We can write the score as

$$S(i, j) = \max_{k=1...m}[S(k, j - 1) + v(i, j, k)], \tag{1}$$

where i is the given state and $v(i, j, k)$ is the value obtained from moving from state k to state i at decision j. The best series of decisions is then found by going backwards, taking the state at each decision i that corresponding to the highest score.

In this most general form, the algorithm is quite expensive. The cost on the order of $n * m * m$. For a large number of states the problem quickly becomes impractical. The easiest way to reduce the cost is to limit the number of states that we must search when moving from one decision to another.

For the interactive picking problem we are looking for the best path through a series of points in 2-D. In this example I am requiring that the solution is single valued along one axis (e.g. only one velocity at each time sample). I am looking for the best path between my first and last picks along the single value axis. I form an initial path by linear interpolation between the selected points. I am then going to limit my search space so it is no more than x points away from the linear path (limit the possible states of k). Figure 1 demonstrates this concept. The left panel shows a semblance scan along with three selected points. The right panel shows the semblance extracted along the path represented by the three points.

In term of equation (1) we have n decisions (the number of time samples between the first selected point and the last selected point) with m states ($2 * d + 1$ in size where d is the search distance from the initial linear path.) Generally v takes the form

$$v(i, j, k) = m(i, k) - p(j, k), \tag{2}$$

where m is the semblance at a given location and p is a scalar that punishes jumps in the selected semblance path. What is used for equation 2 can have dramatic effects on the solution. For the applications discussed in this paper I defined p so that a single sample jump was mildly punished with rapidly increasing penalty with larger gaps. This has the effect of tending to create smooth solutions.

To further create a smooth path I only search in the range $abs(j - k) < 5$. Finally to force the user selected points to be honored I modify m in a manner similar to Harlan (2001). I add large values to m which has the effect of forcing the solution through these points. Figure 2a shows the generated score matrix from the data

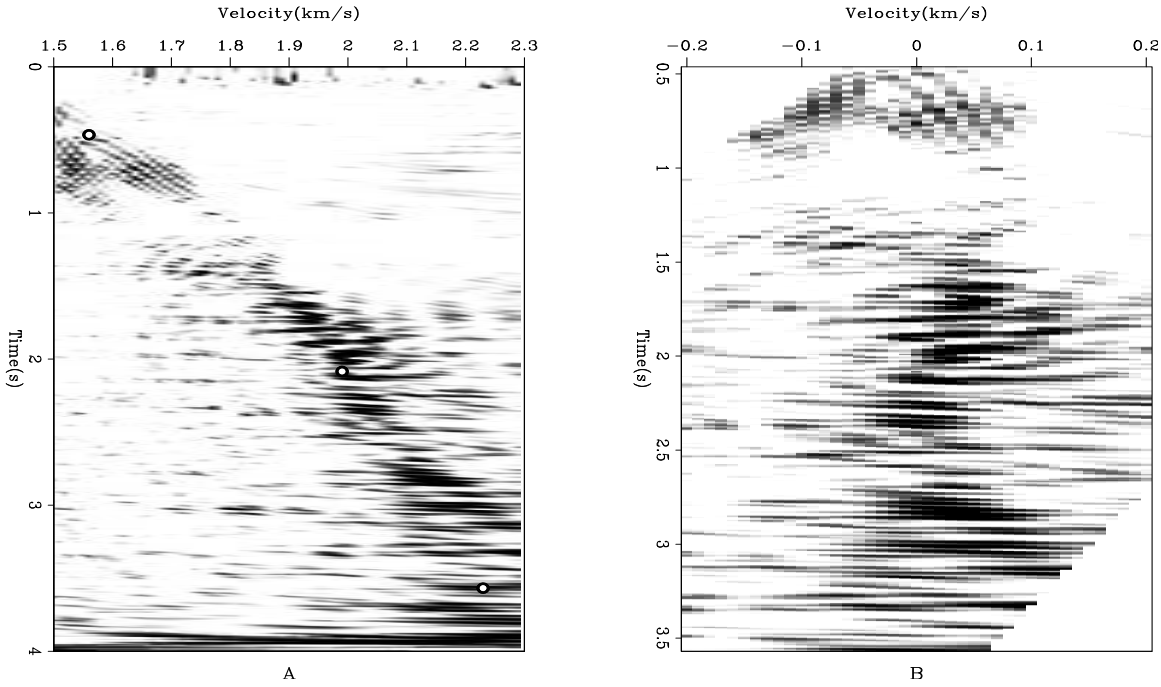


Figure 1: The left panel shows a semblance gather overlain by three picks. The right panel is the result of linearly interpolating between the picks and extracting semblance along the resulting line. [**ER**]

shown in Figure 1b overlain by the traced back path through these points. The path is calculated by finding the maximum in the first row and then searching for the maximum within some range in the next row, proceeding to the bottom of the score matrix. Figure 2b shows the path on the original data

Reflectors can also be auto-picked by this method. In the case of picking reflectors, m becomes the correlation of the data along with an initial linear path. Figure 3 demonstrates the concept. The left panel of Figure 3 show the data with a set of four points selected. The right panel of Figure 3 shows the path picked by the algorithm.

LLOYD

The concept of quantization originates in the field of electrical engineering. The basic idea behind quantization is to describe a continuous function, or one with a large number of samples, by a few representative values. Let x denote the input signal and $\hat{x} = Q(x)$ denote quantized values, where $Q(\cdot)$ is the quantizer mapping function. There will certainly be a distortion if we use $\hat{x}$ to represent x. In the least-square sense, the distortion can be measured by

$$D = \sum_{i}^{n} (x - Q(x))^2. \tag{3}$$

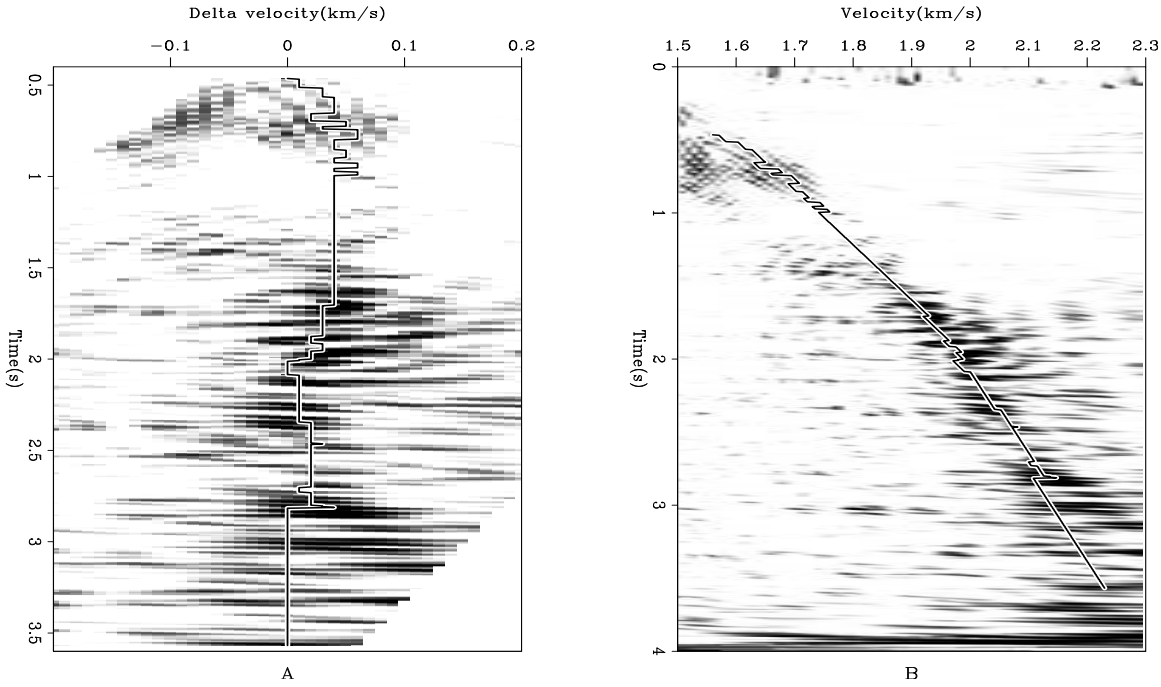


Figure 2: The left panel (A) shows the score matrix calculated using equation 1 overlain by the maximum tracked path. The right panel, B, shows the path overlain on the original semblance display. [**ER**]

Consider the situation with L quantizes $\hat{x} = (\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_L)$. Let the corresponding quantization intervals be

$$T_i = (a_{i-1}, a_i), i = 1, 2, \ldots, L, \tag{4}$$

where $a_0 = min(x)$ and $a_L = max(x)$. The distortion function then becomes

$$D = \sum_{i=1}^{L} \sum_{x=a_{i-1}}^{a_i} P(x)(x - \hat{x}_i)^2, \tag{5}$$

where $P(x)$ is the discrete version of the probability density function, or normalized histogram ($\sum_x P(x) = 1$). To minimize the distortion function D, we take derivatives of equation (5) with respect to $\hat{x}_i$, a_i and set them equal to zero, leading to the following conditions for the optimum quantizers $\hat{x}_i$ and quantization interval boundaries $\hat{a}_i$:

$$\hat{a}_i = \frac{\hat{x}_i + \hat{x}_{i+1}}{2}, \tag{6}$$

$$\hat{x}_i = \frac{\sum_{x=\hat{a}_{i-1}}^{\hat{a}_i} P(x)x}{\sum_{x=\hat{a}_{i-1}}^{\hat{a}_i} P(x)}. \tag{7}$$

A way to solve this coupled set of nonlinear equations is to first generate an initial set $\{x_1, x_2, \ldots, x_L\}$, then apply equations (6) and (7) alternately until convergence

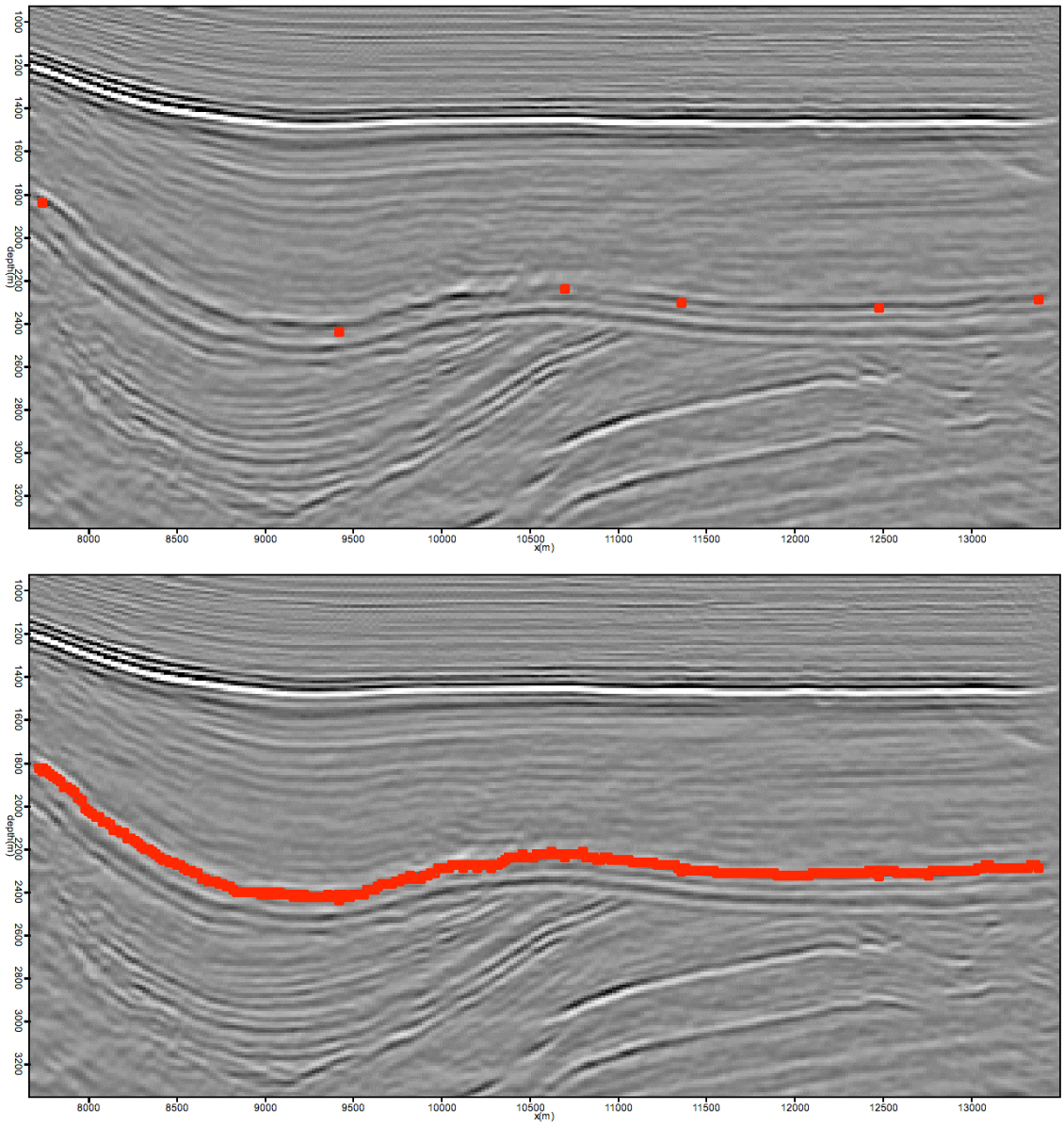


Figure 3: The left panel shows the original with four seed points selected. The right panel shows the result of using the Viterbi algorithm to pick the reflector. [**NR**]

is obtained. This iteration is well known as the Lloyd-Max quantization algorithm (LMQ). A common modification is to form

$$D(i) = \sum_{x=a_{i-1}}^{a_i} P(x)(x - \hat{x}_i)^2, \tag{8}$$

and to remove a_i where the distortion is small and possibly add as in regions where the distortion is large. The resulting a locations is often much smaller than the initial set of values.

The LMQ scheme is designed to find the best representation of a distribution, which is not what I am trying to do in this instance. Instead I am trying to the achieve the representation of $y(x)$ with as few x_i, y_i points as possible. The twist on the standard LMQ scheme is the replacement of $P(x)$ in equation 5. Instead of being the probability density function I construct an error from a background piece-wise linear function. I first construct $z(x)$ by linear interpolating between x_i, y_i samples. I then calculate $d(x) = y(x) - z(x) + min(y(x))$, the error from the piecewise linear background. Figure 4 demonstrates the methodology. Figure 4a shows a curve with '*' the initial x_i, y_i points and the resulting $z(x)$ function. Figure 4b shows the $d(x)$ function constructed from $y(x)$ and $z(x)$. We now have something that is approximating the shape of a probability density function except that it can be positive or negative. To get around this problem I first

$$sn = \sum_{i=x_{i-1}}^{x_i} d(i), \tag{9}$$

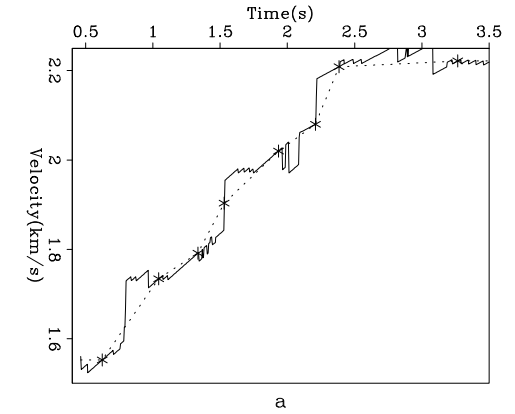
then if sn is positive I define $P(x)$,

$$P(x_{i-1}..x_i) = d(x_{i-1}...x_i) + min(d(x_{i-1}...x_i)). \tag{10}$$

If $sn < 0$ I define

$$P(x_{i-1}..x_i) = -d(x_{i-1}...x_i) - max(d(x_{i-1}...x_i)). \tag{11}$$

As a result $P(x_{i-1}..x_i)$ is always positive. Flipping the signs does not violate the LMQ concept. What equation 7 is attempting to do is a *local* center of mass calculation. By applying equation 10 or 11 we are transforming our coordinate system to obtain an accurate center of mass calculation. How accurate the curve is represented is determined by the number of a_i terms. In practice it is best to start with a dense representation of a_i to avoid local minima and then use the fitting criteria of equation 8 to eliminate points in regions with small deviations. Figure 5 demonstrates this concept. The solid curve in Figure 5 is the original function. The three dashed curves show different deviation criteria. With increasing accuracy an increasing number of points are needed to represent the curve. In this example 2, 9, 28 and points are used.



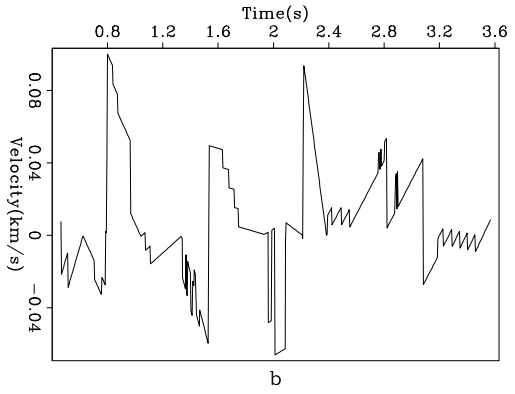
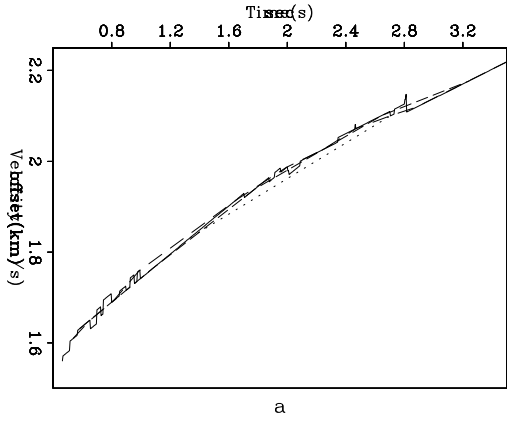


Figure 4: Panel (a) shows the original curve (solid line); an initial set of a value, asterisks; and the background, dashed, curve z. Panel (b) shows the deviation d from the piece-wise linear background. **[ER]**



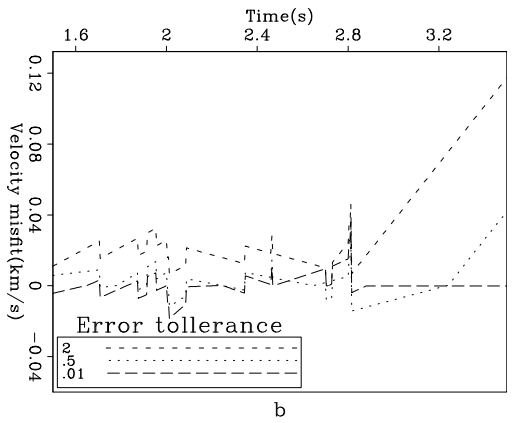


Figure 5: The effect of modifying the deviation criteria. In panel (a) the solid curve in Figure 5 is the original function. The three dashed curves show three different deviation criteria. The closer the fit to the original curve the more points that are needed for an accurate representation. Panel (b) shows the error in the fitting functions. **[ER]**

Extensions

As demonstrated in Clapp (2006); Tang and Clapp (2006) Lloyd's algorithm is easily extended to multi-dimensions. A dense two or three dimensional volume of picks could be reduced to a few hundred or thousand points enabling a relatively easy QC process.

CONCLUSION

Dynamic programming can be effective in automatically picking surfaces and semblance. The method, in current form, is limited to 2-D, limits its applicability to the surface picking problem. Lloyd's algorithm offers a way to QC automatically picked volumes. By replacing the probability distribution function with an error misfit function, Lloyd's algorithm can be used to effectively characterize a dense series to a sparser set of picks while maintaining its essential character.

REFERENCES

Clapp, R. G., 2006, A modified lloyd algorithm for characterizing vector fields: SEP-Report, **124**, 249–256.

Harlan, W., 2001, Automatic moveout picking: WWW. (http://billharlan.com/pub/paper/autopick.pdf).

Hosom, J.-P., R. Cole, and M. Fanty, 1999, Speech recognition using neural networks at the center for spoken language understanding: http://cslu.cse.ogi.edu/tutordemos/nnet_recog/recog.html.

Kruse, C., 1988, A new method of nonlinear signal correlation using the instantaneous spectrum: 68th Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 1321–1323.

Liner, C. L. and R. G. Clapp, 2004, Nonlinear pairwise alignment of seismic traces: Geophysics, **69**, 1552–1559.

Lloyd, S. P., 1982, Least squares quantization in pcm: IEEE Transactions on Information Theory, 127–135.

Needleman, S. and C. Wunsch, 1970, A general method applicable to the search for similarities in the amino acid sequence of two proteins: J. Mol. Biol., **48**, 443–453.

Tang, Y. and R. G. Clapp, 2006, Lloyd's algorithm for selecting reference anisotropic parameters during wavefield extrapolation: SEP-Report, **124**, 257–270.

Zhang, L., 1991, Automatic picking and its applications: SEP-Report, **70**, 275–292.