Methods for feature extraction from time series data

Steven W. Morris
morris@ics.uci.edu

Technical Report 92-17

January 17, 1992

°This research was supported in part by Douglas Aircraft Company and the University of California through MICRO project 90–117.

Contents

1 Introduction

In this paper we review some techniques that can be used for the extraction of features from time series data. In particular we report some results obtained in using two feature extraction methods. We have used both methods to extract descriptions of peaks in A-scan ultrasound readings of metal parts as part of a joint research effort with Douglas Aircraft Company and McDonnel Douglas Research Laboratories to automate the detection of cracks. The first feature extraction method, which we call Filtered Scale-space, is an extension of techniques used in Witkin's Scale-space method for the qualitative description of data. The second is a method of feature extraction developed at the University of California, Irvine. This method extracts primitive features by analyzing the amplitude variations in the time series data. From these primitives, an FSA is used to synthesize simple peak features. These peak features can then be used to synthesize more complicated features that are at an appropriate level of abstraction for problem solving. For example, in ultrasonic diagnosis of metal parts, the features in the ultrasound signal that are diagnostic might not simply be single peaks but instead a contiguous sequence of peaks.

2 Nondestructive Evaluation using Ultrasound

Nondestructive evaluation (NDE) employs techniques to inspect materials without damaging them. One of the techniques most often used by aircraft manufacturers and operators is ultrasonic testing. We have been researching feature extraction methods that can be used to generate qualitative descriptions of ultrasound readings such as that shown in Figure 1. This research has been part of a program of joint research with Douglas Aircraft Company and McDonnel Douglas Research Laboratories that aims at automating the detection of flaws in aircraft parts. Before describing the feature extraction methods, we present a little background on the ultrasonic testing domain.

Ultrasonic testing utilizes sound waves in the 1 MHz to 25 MHz frequency range to measure the thickness of a material or to examine the internal structure of a material for possible defects such as voids, delaminations, and cracks. By transmitting a pulse of sound through the material and examining the amount of sound energy that is transmitted or reflected, it is possible to make determinations about the internal structure of the material (Gallagher, Giessler, Berens & Engle, 1984; McMaster, 1959). In a pulse-echo "A-scan" setup, portions of the sound pulse reflect off material discontinuities in the test piece. These reflections are picked up by a transducer coupled to the surface of the test piece and transformed into an electrical signal that can be displayed on a CRT or digitized and recorded.

Figure 1 shows a digitized A-scan signal from an aluminum block that is one of a set of nine blocks used as a reference standard in NDE tests. The reading shows reflections from the front and back surfaces of the block, as well as a reflection from a manufactured defect that exists at a known location inside the block. This defect consists of the surface of the bottom of a hole carefully drilled from the back wall of the block. The horizontal axis of the graph indicates the time of flight of the sound reflections. Figure 2 shows schematically one

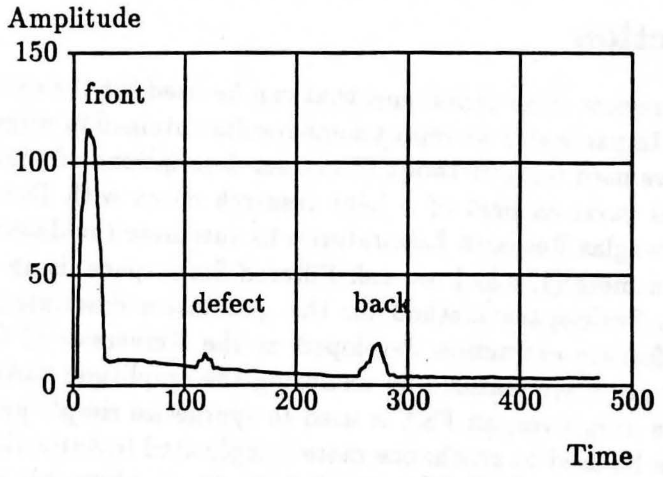

Figure 1: Typical A-scan of defective part

of the test blocks and the ultrasonic test setup.

The approach we are taking is one in which we extract those primitive features that are diagnostic with respect to a given domain theory. In our case of ultrasonic diagnosis, the kind of domain theory we have in mind is one that can be used to reason about the basic geometry of metal blocks (surfaces, spatial extent) and the motion of objects (reflected and transmitted sound pulses). Such a theory is to be used to explain the existence of the sound reflections that are observed in an ultrasonic reading by reasoning that each particular reflection arises either from the presence of a known surface, or from the presence of a hypothesized defect in the metal block.

The initial task we set ourselves was to process a reading such as that in Figure 1 and extract features that describe the 'front,' 'back,' and 'defect' reflections. While the reading in Figure 1 shows a relatively strong defect reflection, readings from other test blocks have defect reflections that are smaller. Figure 3 shows such a reading in which the defect reflection is the small peak located at time 190. Figure 4 shows a portion of this reading with a magnified vertical scale. This figure shows the defect more clearly and it also shows the kind of low amplitude, fine structure present in these readings.

3 Some standard methods used in feature extraction

This section discusses feature extraction some established methods that can be used in feature extraction from time series data. In particular, we discuss an augmentation of the Scale-space method, that we call Filtered Scale-space, that we have tested for use in the ultrasonic test block domain.

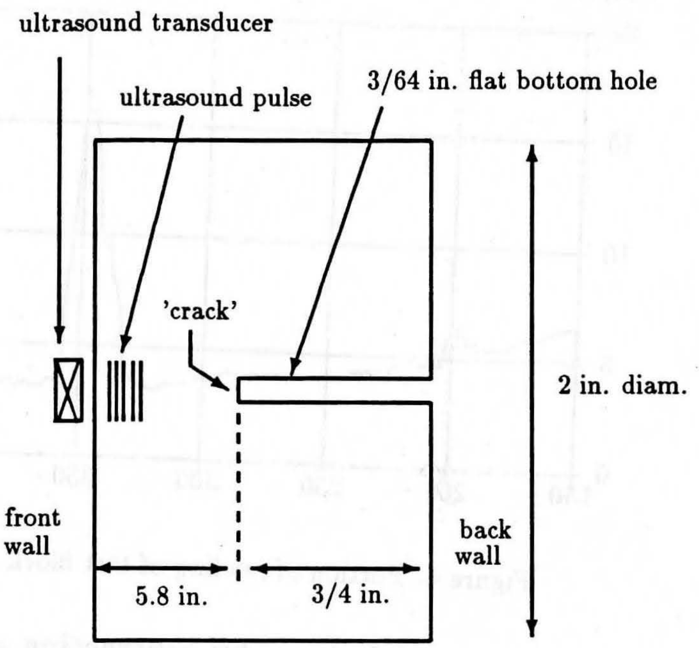

Figure 2: Ultrasonic test setup of an aluminum test block

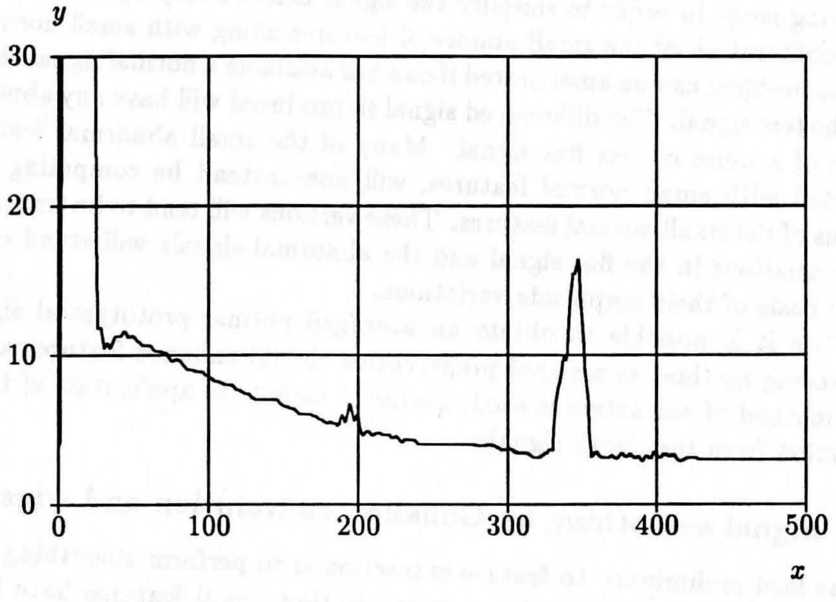

Figure 3: Reading of test block D42C.38

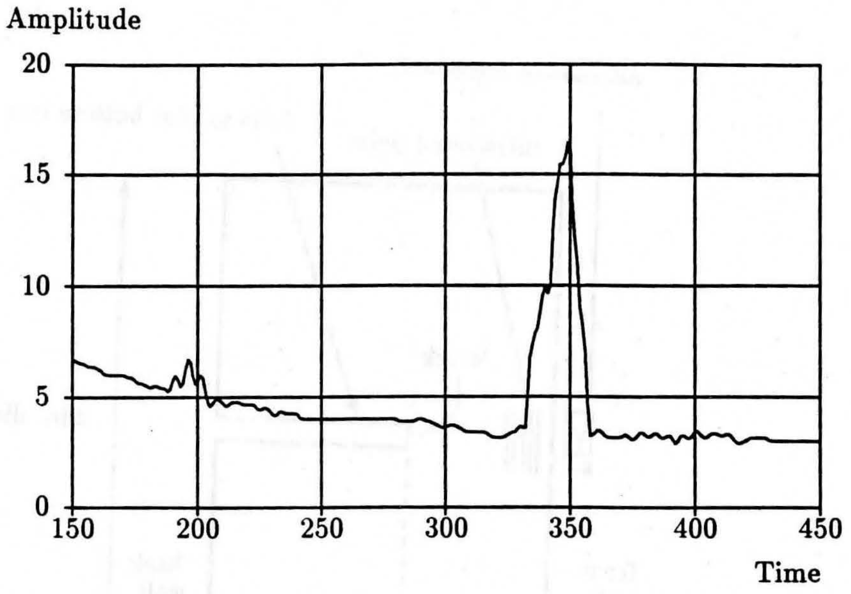

Figure 4: Portion of reading of test block D42C.38

3.1 Detecting abnormal features by subtracting a normal signal

In doing diagnosis one wants to detect in signals those features that don't appear in signals obtained under normal circumstances. However, sometimes the size of an abnormal feature is close to the size of normal features in the signal. Any approach that uses a preprocessing smoothing stage in order to simplify the signal before applying feature extraction will result in the obliteration of the small abnormal features along with small normal features.

This problem can be ameliorated if one has available a normal signal that can be subtracted from the test signal. The differenced signal so produced will have any abnormal features sitting on top of a more or less flat signal. Many of the small abnormal features that previously competed with small normal features, will now instead be competing with the differenced versions of the small normal features. These versions will tend to be transformed into relatively small variations in the flat signal and the abnormal signals will stand out and be detectable on the basis of their amplitude variations.

When it is possible to obtain an averaged normal prototypical signal, then using this differencing method as a signal preprocessor should enhance feature extraction regardless of what method of extraction is used. Section 5 shows the application of this method to feature extraction from test block signals.

3.2 Signal smoothing by Gaussian convolution and edge detection

A standard preliminary to feature extraction is to perform smoothing in order to produce a simplified signal in which noise or uninteresting, small features have been suppressed while larger, interesting features have been preserved (although perhaps somewhat modified) (Chen, 1982).

A great variety of filters exist for this purpose. One kind of filter that has been found particularly useful in vision applications is the discrete convolution with the Gaussian function and its derivatives. In vision applications, an "edge" (significant change in the signal) is a primitive feature that is diagnostic for the task of image understanding. This is because edges are reliably associated with "discontinuities in physical properties, such as depth, surface orientation, or reflectance properties" of the objects that have been imaged (Ullman, 1986). In vision applications, the convolution with the second derivative of a Guassian has been found to have optimal properties for the location of edges (Marr & Hildreth, 1980).

Just as edges are diagnostic in vision, the amplitude variations present in ultrasonic signals can also be analyzed in terms of edges. In particular, the location of an ultrasound peak can be characterized by the inflections points (edges) that bound it on the left and right. Therefore, one approach to feature extraction of ultrasonic signals that we have investigated is Witkin's Scale-space method, which is founded upon edge detection using Gaussian filtering.[1]

3.3 Scale-space method

The first feature extraction method we tried is based on a technique used in Witkin's Scale-space method (Witkin, 1983).

3.3.1 Description of method

The Scale-space method works by smoothing the signal with a sequence of filters that provide increasing degrees of smoothing, σ, and by plotting the zero-crossings of the second derivative of each smoothed signal as a function of their location and of the strength of the filter at which they are produced. The parameter that controls the degree of smoothing, σ, is the standard deviation of the Gaussian filter used, and is a measure of the width of the Gaussian. Figure 5 shows an example of a Scale-space, zero-crossings plot for one of the test block ultrasound readings.

The Scale-space method goes on to derive from the Scale-space plot a data structure called an "interval tree" that provides a means of producing a qualitative description of the data based upon picking the best σ at which to locate edges within localized segments of the data.

In Figure 5 the zero-crossings can be arranged into lines rising vertically through the Scale-space. Each line, called a "contour," is associated with a single inflection point as it survives across successive degrees of smoothing. In general, contours vanish in pairs corresponding to two inflection points approaching and meeting each other as the common peak that they bound is smoothed out. These two contours can be traced down to the time axis where they identify the two inflection points in the unsmoothed data. Thus, pairs of contours allow the identification of a feature. Without seeing this merging of contours, one might for example, try filtering at a single σ, find the zero-crossings, but then inappropriately pair them up. This is because successive zero crossings often do not belong to the same feature. That is, there

[1]This approach was originally recommended to us by Deepak Kulkarni at NASA Ames.

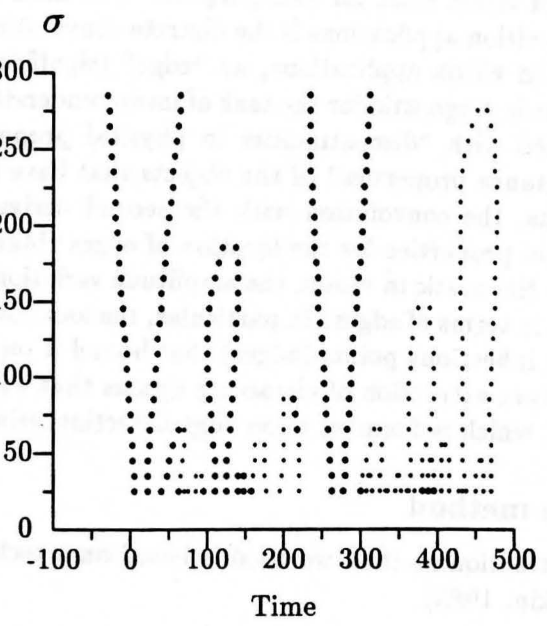

Figure 5: Scale-space contours for a $D35$ signal

are pairs of zero crossings representing smaller features, that are nested within a pair of zero crossings which represents a larger feature.

3.3.2 Implementation

We used code written at NASA-Ames by Kiriakos Kutulakos and subsequently modified by Steve Morris at UC Irvine.

3.3.3 Our experience with the Scale-space plot

When we used this method on ultrasound readings of test blocks, we found that 'undesirable' zero-crossing contours were produced. These were contours produced by edges associated with very low amplitude variations in the data. For example, in Figure 5, in addition to the three robust contour pairs corresponding to the front surface, defect, and back surface of the block (the six contours on the left that survive above $\sigma = 175$), there is also an undesired long-lived contour pair on the far right. These two contours are generated by inflection points in a region of the data containing relatively small variations in amplitude. On the other hand, the relatively large amplitude peak associated with the defect does not survive as long; its inflection points are smoothed out by $\sigma = 190$.

We had hoped that the contours of the small amplitude features would die before those of the relatively large features of interest. We could have then used the Scale-space plots to

pick a single σ at which the surviving contours could be traced back down to the $\sigma = 0$ axis to determine the feature edge locations in the unsmoothed data.

The undesired contours showed this would not work. This phenomenon perhaps indicates that the Scale-space technique does not provide global comparative information about feature size throughout a data set, but instead provides only local comparative information. In the 'full blown' Scale-space technique, such local comparative information is encoded by an "interval tree." This tree is then used to determine the σ's that provide optimal local description of the data. But we are not using this full-blown Scale-space method. We are only attempting to borrow the idea of the Scale-space plot of zero-crossings to determine significant edges, where 'significant' for us means 'associated with a large feature.' Besides, we are not interested in locally optimal descriptions of a reading. We instead want to globally ignore what we consider uninteresting or noisy features and extract only the interesting, big features. The interval tree approach would instead simply give us locally optimal descriptions of all the features in the reading, noise and all.

3.4 Filtered Scale-space

We developed a modification to the Scale-space plot that appears to solve the problem of the undesirable contours. Looking at plots of the second derivative of the data at any particular σ, we noticed that, for the significant features, the third derivative at the zero-crossing points were about an order of magnitude or greater than the third derivative at the 'undesirable' zero-crossings. This suggested a test, using a threshold on the third derivative at the zero-crossings, to filter out of the Scale-space plot the zero-crossings of the undesirable features. For the test block signals, the test decided upon was: for a given scale σ, if the absolute value of the slope of a zero-crossing is less than 0.005 of the highest absolute slope of all the zero-crossings, then eliminate the zero-crossing. The value 0.005 was chosen because it works for as many of the nine test blocks as possible. For example, the "dark" zero-crossings in Figure 5 are those that survived this test, while the "light" zero-crossings are those that did not survive. For more details, see (Amirfathi, Morris, O'Rorke, Bond, & St. Clair, 1991) and (O'Rorke, Morris, Amirfathi, Bond, & St. Clair, 1991).

The performance of this test was that it effectively killed the 'undesirable' contours, and allowed the choice of a single contour-sampling-σ that would extract all the significant features for seven of the nine blocks. For each of the other two blocks, the zero-crossing test killed large parts of one of the two contours associated with the crack, so that at the sampling σ, it is either the case that only one of the edges of the crack is seen, or the case that the contour for one edge gets obliterated in its mid-σ range and therefore can't be traced back to the $\sigma = 0$ axis.

The errors on these two blocks are due to the fact that their crack signals are small, or similar in size to a non-significant feature that is present in the RMS data for each of the blocks. This non-significant feature is a small 'hill' that immediately follows the front wall feature, and which apparently is due to a near-field effect inherent in the ultrasound transducer and setup used. For the reading D42C.38, the crack peak is similar in size to this small hill, while for the reading D42C.31, the crack peak is 'buried' in (obscured by) this small hill. The existence of this 'competing' hill limits the size of resolvable significant features. (But see Section 5.)

3.5 E-filtering

A phenomenon that occurs with Gaussian smoothing such as in Scale-space is that the location of edges (inflection points) migrate within the smoothed versions of the data. This is manifested in the Scale-space plot as curved contours. A technique to lessen this migration is the E-filter, introduced in (Moore & Parker, 1973) and generalized in (Saito & Cunningham, 1990). In this technique, prior to smoothing the data, a non-linear transformation is applied to the data. This transformation modifies the x values of the data's x-y points and spreads out the data set by varying amounts. The Gaussian (or other linear filter) is then applied, after which an inverse transformation is applied to the smoothed data that restores the original x-values. So long as a minimum aspect ratio of height to width of feature exists for the significant features, this method will reduce the amount of edge migration in the smoothed data. The minimum aspect ratio can be achieved by multiplying the y-values of the unsmoothed data by a constant prior to E-filtering.

Saito & Cunningham (1990) introduced a generalized E-filter in which the amount of data spreading (the degree of non-linearity) is controlled by a parameter. They claim that their generalized E-filter solves several problems with the Scale-space technique:

> In practice, there are some difficulties in implementing the scale-space filter ... Computation of the second derivatives of sampled data is a noisy process. In most of our applications, we have looked at extreme points of the first derivatives rather than the zero crossings of the second derivatives. Edges are defined by extreme values of the first derivative greater than some threshold value.
>
> A more severe problem is tracking the edge: as the Gaussian widths decrease, more and more edges are detected, and identification of the significant structure becomes problematic. Witkin [8] ... devotes a significant amount of time to developing a hierarchical tree to assist in following the edge. ... as the Gaussian width increases, the edge positions migrate some distance from their original location.
>
> ... scale-space filtering will suffer from edge migration due to the coherent interference of neighboring structures.
>
> With the use of the E-filter, it is therefore possible to simply filter with a large width and obtain the placement of the edge positions directly, without having to track the structures to small Gaussian widths. This is a significant saving in terms of computational effort.
>
> ... the generalized E-filter also solves another difficulty found with scale-space filtering, ... the "masking effect", and is related to the loss of high-frequency information for large values of the Gaussian width. In Fig. 2 we see an illustration of the problem. The two structures marked "b" are separated by a relatively small gap. Standard linear filtering, as used in scale-space, cannot resolve them, and

only one structure is identified. When the edge detection is based on the E-filter approach, the two structures remain separated, even for large Gaussian widths ...

This technique seems to present the possibility that a single second derivative Gaussian filter can be used, with the E-transformation, to locate zero-crossings (edges) in the smoothed data. Because of small edge migration, we could then 'read off' the edge locations directly from the smoothed data. This would presumably avoid the multiple filterings of the Scale-space technique and thus save computation.

It would seem that this single filter technique suffices if all you're interested in is locating edges. But if you are primarily interested in locating peaks by finding their bounding edges, then using a single filter to find zero-crossings might not be a robust technique in light of the possibility of not being able to properly identify which zero-crossings should be paired together (see above) without using the Scale-space plot. Perhaps a single filter approach, combined with determining the edge pairings by looking directly at the smoothed data curve, might work. Regardless, E-transformation to reduce edge migration is a good technique to keep in mind.

4 Feature extraction by interpretation of a signal's amplitude changes.

The Scale-space method depends on computing several filtered versions of the signal in order to form contours. The need to look at contours in the Scale-space method arises from the fact that zero-crossings, and the edges they represent, provide very local information about the signal. In order to realize which edges to pair (so that, for instance, one does not pair an edge of a large feature with that of a small feature superimposed upon it) one must observe how the contours vanish as smoothing increases.

An alternate approach that can discriminate small features from large is to actually look at the amplitude variations of the signal. In this way, one can locate segments of the signal in which the overall amplitude change is large while ignoring the small amplitude ups and downs within that segment. In this section we describe a feature extraction algorithm that does this. It is based upon the analysis of a single signal rather than analyzing a sequence of signals as in the Scale-space method. It therefore has the potential to be quite a bit less computationally expensive than the Scale-space method.

4.1 Overview of the feature extraction method.

The feature extraction method consists of three stages. See Figure 6. The first stage runs an algorithm that receives the signal as input and produces a sequence of primitive features as output. Each primitive describes a segment of the signal, and the sequence of primitives partitions the signal. The primitives that we have chosen are : *incline*, *decline*, *flat*, and *insignificant*. For example, an incline is a segment of the signal that is described as an increase in amplitude over a particular range of the independent variable, and the average slope of this increase meets a certain minimum value as indicated by a parameter of the

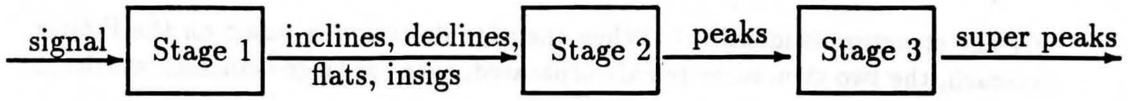

Figure 6: Overview of feature extraction method.

Stage 1 algorithm. Section 4.2 describes the four primitives and how the Stage 1 algorithm works.

Stage 2 of the method analyzes the output of Stage 1 and identifies *peaks* in the signal. This stage uses a simple finite state automaton to find sequences of the primitives that correspond to peaks in the signal. A simple example would be two inclines followed by a decline. Section 4.3 describes this process.

In the ultrasonic diagnosis of cracks a meaningful feature is generally not as simple as a single peak. For instance, a reflection from a crack might be described in terms of a sequence of three peaks. We therefore have a third stage that analyzes the peak information produced by Stage 2 to synthesize 'super' features that are at the appropriate level for applying diagnostic knowledge. Whereas as Stage 1 and Stage 2 result in a signal description couched in a primitive language of peaks that is general and domain independent, Stage 3 is the level at which domain knowledge about the kind of complex 'signatures' produced in the signal by the underlying processes of interest (such as sound reflecting from cracks) can be brought to bear. Often these signature features are more complex than simply a single peak. Rather they may comprise a sequence of peaks, perhaps having amplitude or width relations to each other. Section 4.4 discusses this last stage of the feature extraction method which culminates in the feature descriptions of interest to the diagnostic process.

4.2 Stage 1 : Extracting primitive features.

The Stage 1 algorithm processes the signal which is described as a time series of amplitudes changes. The algorithm partitions the signal into a sequence of primitive feature descriptions. Each primitive describes a time segment of the signal by a symbolic type (eg., an 'incline') plus one or more numeric values that characterize the segment. The algorithm basically makes a pass through the input data from left (early times) to right (later times) recognizing primitive features as it does so. In the process of recognizing and describing a particular feature, the algorithm can backtrack and modify or even delete earlier features. This is done in order to arrive at a more optimal description of two or more adjacent features. We next describe the Stage 1 algorithm in more detail.

4.2.1 Input and output of the algorithm.

One of the inputs to the algorithm is a representation of a time series function in terms of its "ups" and "downs." That is, the function is described as an initial value and a sequence of subsequent amplitude changes. For example : $y = 3.2$ at $t = 1.2$; $\Delta y = 3.24$ from $t = 1.2$ to $t = 1.4$; $\Delta y = -1.7$ from $t = 1.4$ to $t = 1.7$; etc. For convenience, I will refer to these amplitude changes as "deltas."

The other inputs to the algorithm are several numeric parameters that control how features are extracted. These parameters are described in section 4.2.2.

The output of the algorithm is a description of the signal in terms of a predefined set of primitive features. If desired, these primitive features can serve as input to other algorithms that synthesize them into more complicated features. For example, the algorithm as currently implemented produces a description of a signal in terms of the following primitive features : ***inclines***, ***declines***, ***flats***, and ***insignificants***. The feature type ***insignificant*** is a catch-all used to describe any sequence of deltas that is not described by the other three feature types. The three other feature types are considered to be "significant" features. The user of the feature extraction method tries to set the numeric parameters so that the features deemed significant are extracted while all other deltas sequences get classified as ***insignificants***. Having this catch-all feature type guarantees that every delta is incorporated into some feature.

The algorithm analyzes the sequence of deltas and aggregates contiguous subsequences of them into features. Upon termination it outputs a description of the signal as a sequence of features, the k-th feature being described by the tuple $(f_k, tbeg_k, tend_k, attr1_k, \ldots, attrj_k)$, where f_k is one of the feature types, $tbeg_k$ and $tend_k$ are the beginning and ending times of the feature, and $attr1_k, \ldots, attrj_k$ is a sequence of j different attribute values that are recorded for the feature. In our initial implementation, we record only one attribute for each feature, namely the change in amplitude of the function in the interval $[t_{beg}, t_{end}]$. Because every delta is incoporated into a feature, the output of the algorithm partitions the time series function into a feature sequence : $(f_1, tbeg_1, tend_1, attr1_1), \ldots (f_k, tbeg_n, tend_n, attr1_n)$, where $tend_k = tbeg_{k+1}$.

4.2.2 Description of the feature types and the five numerical parameters.

We next describe in more detail what constitutes an incline, a decline, a flat and an insignificant. There are five numeric inputs to the algorithm all of which are positive floating point numbers : *sig.ampl.change*, *sig.slope*, *sig.flat.width*, *sig.flat.slope*, *max.slope.change*. For any particular run of the Stage 1 algorithm these numeric parameters control how the time series function is carved up into the primitive features. The fifth parameter, *max.slope.change*, will be explained later after the Stage 1 algorithm has been described in the next section. The first four parameters define feature classes as follows.

An *incline* is a segment of the signal in which the net amplitude change, Δy, increases by at least the value + *sig.ampl.change* **and** whose average slope, $\Delta y/(tend - tbeg)$, has at least the value - *sig.slope*. In other words, an incline is a sufficiently large and sufficiently steep increase in the signal.

Similarly, a *decline* is a segment of the signal in which the net amplitude change, Δy, decreases by at least the value - *sig.ampl.change* **and** whose average slope, $\Delta y/(tend - tbeg)$, has at least the value - *sig.slope*. In other words, an decline is a sufficiently large and sufficiently steep decrease in the signal.

A *flat* is a segment of the signal whose width, *tend–tbeg*, has at least the value *sig.flat.width* **and** whose average slope, $\Delta y/(tend - tbeg)$ is no greater than + *sig.flat.slope* and no less than

– *sig.flat.slope*. In other words, a flat is a signal segment that is sufficiently 'wide' and sufficiently 'horizontal'.

An *insignificant* is any segment of the signal that does not get classified as an incline, a decline, or a flat.

Figure 14 shows an example of a primitive description of a signal segment that contains two peaks (as defined above) bracketed by two flats. To produce this graph, each primitive (incline, decline, flat, insignificant) is graphed as a straight line with the appropriate slope. In this figure, inclines are labeled with "i", declines with "d", flats with "f", and insignificants with "u".

By judiciously setting the parameters, one can control what kinds of variations in the signal are recognized as features and what kinds of variations are ignored. For example, the parameters *sig.ampl.change* and *sig.slope* control what signal segments are recognized as inclines and declines. Thus, by making *sig.ampl.change* sufficiently large, a segment of the signal which trends upward and which has "superimposed" upon it some low amplitude ups and downs, can be classified as an incline feature while the variations comprising the superimposed 'noise' are not recognized as features. By making the value of *sig.slope* sufficiently large, increases or decreases in the signal that are shallow, even though they are large, can be classified as insignificants. For a final example, by making the value of *sig.flat.width* very large one can effectively eliminate the feature class of flats.

4.2.3 Description of the algorithm.

The Stage 1 algorithm works by forming features from contiguous sequences of deltas, and then doing some local re-assignment of deltas at the boundaries between features to quasi-optimize the definition of the features. This re-adjustment is done using rules that govern when one type of feature is allowed to steal deltas from another type of feature. For example, after an incline has been formed adjacent to an insignificant, the incline is allowed to "steal" some or all of the insignificant's deltas thus increasing the width of the incline and reducing the width of the insignificant.

There are a number of such "stealing rules" that were arrived at empirically in the course of designing the algorithm. This section describes these rules and how the algorithm works in general.

Some terminology.

In the description of stealing rules, for convenience, we will refer to the stealing feature as the *robber* and the stolen-from feature as the *victim.* A feature *expands* when it adds deltas to itself either on its left or on its right. A feature *shrinks* when it has deltas stolen from it either on its left or on its right. The *amplitude* of a feature is the change in amplitude of the signal from the beginning time of the feature to the ending time. For example, a decline has an amplitude that is a negative number. The *width* of a feature is its ending time minus its beginning time. The *slope* of a feature is its amplitude divided by its width. The *initial slope* of a feature is its slope at the point that the feature is initially formed, before the feature has expanded or shrunk. The *extreme slope* of an incline is a record of the most positive slope the incline has ever had over its history. The *extreme slope* of a decline is a record of the most negative slope the decline has ever had. The *extreme slope* of an incline or a decline is initialized to its initial slope. A *delta sequence* is a set of one or more contiguous deltas in the signal.

Other partial background information : A robber always steals deltas that are adjacent to its current boundary. A robber can steal an adjacent sequence of multiple contiguous deltas. Robbers can expand by stealing on their left or on their right. All of the above is contingent upon meeting all the stealing rules that are applicable to the robber. Whether stealing occurs on the left or on the right depends upon the part of the algorithm being executed at the moment. The algorithm is completely deterministic.

Stealing rules.

Rule G1 : A robber cannot steal a delta and incorporate it into itself if doing so would cause the robber's feature type to change (E.g., an incline cannot absorb a delta and become an insignificant.)

Rule G2 : A victim cannot have a delta stolen from it if doing so would change the victim's feature type. However, it is allowed for a victim to have all of its deltas stolen and thus go out of existence.

Rule ID1 : Inclines and declines can steal from any adjacent feature including features of their same type (e.g., an incline can steal from an incline).

Rule ID2 : Inclines and declines cannot steal a delta sequence that results in reducing the absolute value of the feature's amplitude. I.e., an incline cannot reduce its amplitude and a decline cannot increase its amplitude (go less negative).

Rule F1 : Flats can only steal from flats or from insignificants.

Rule U1 : Insignificants cannot steal deltas.

Rule MSC : This rule constitutes a test that must by passed by inclines and declines whenever they expand or shrink. It does not apply to flats or insignificants. The expansion or shrinking of inclines and declines is governed by the following constraint : A delta sequence may be added (expansion) or taken away (shrinking) only if the resulting change in slope is within $\pm$*max.slope.change* of the extreme slope of the feature. If this test is passed and the new slope is more extreme than the old extreme slope (i.e., more positive in the case of an incline; more negative in the case of a decline), the extreme slope of the feature is reset to this new slope. All of this has the consequence that the slope of an incline or a decline can change, but such changes are bounded by the historically extreme slope of the feature.

The MSC rule and the extreme slope updating provides a means of controlling how accurately the shape of the signal is characterized by the primitive features extracted by Stage 1. An example of this is shown in Figure 7 and Figure 8. These figures show a close-up of the back wall reflection for the *D*38 signal. In Figure 7 the dashed lines show the primitive extracted with the setting *max.slope.change*= 0.7. In Figure 8 are shown the primitives with *max.slope.change*= 0.3.

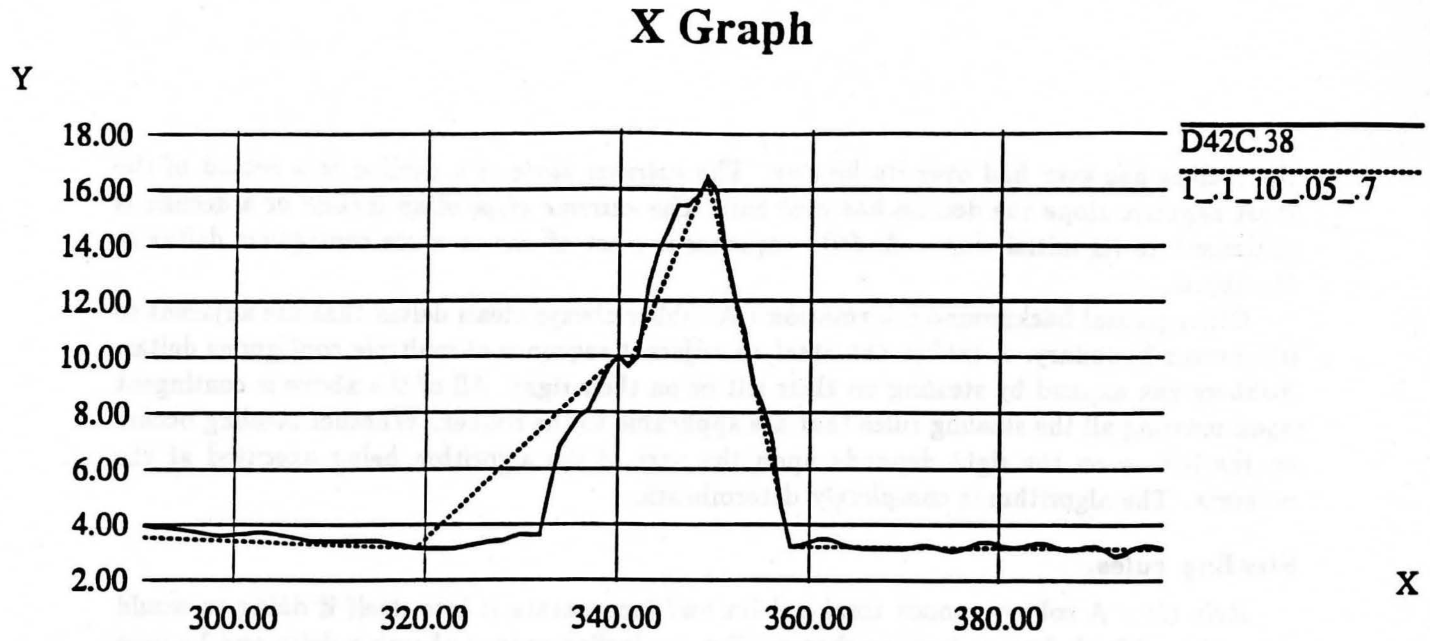

Figure 7: D42C.38 back wall using *max.slope.change* = 0.7

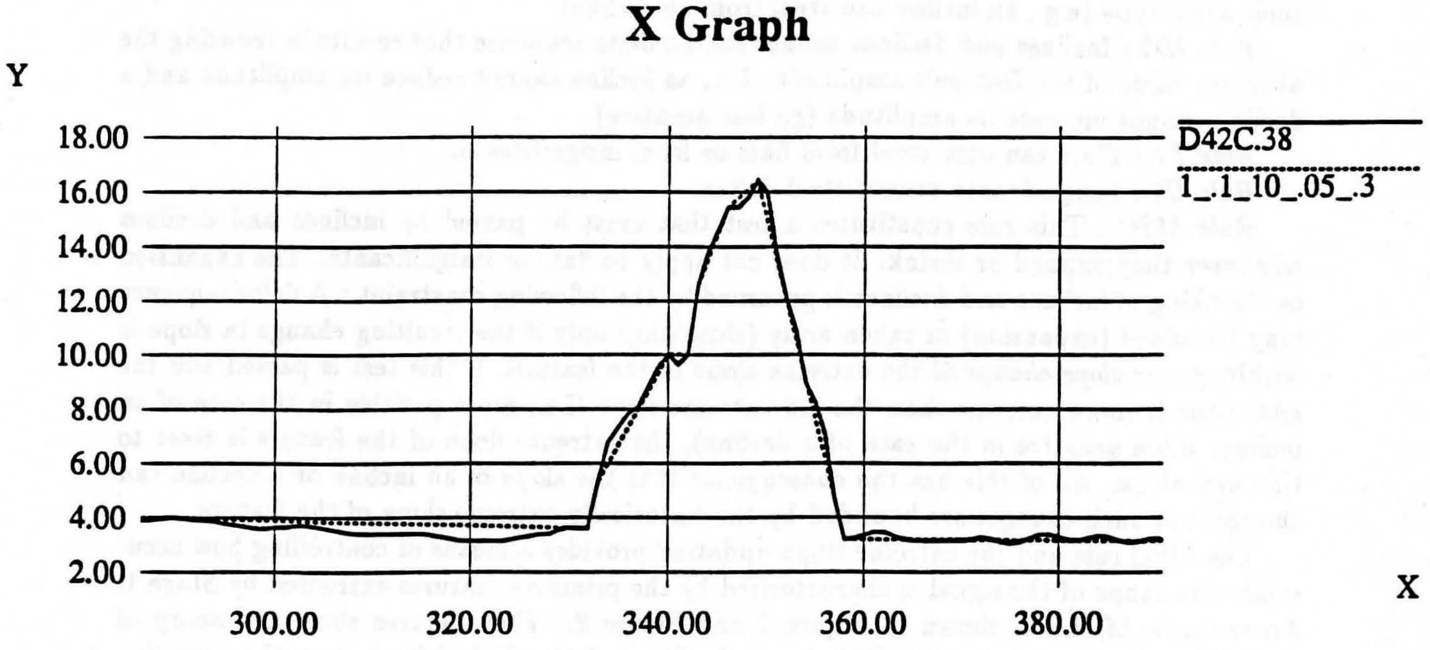

Figure 8: D42C.38 back wall using *max.slope.change* = 0.3

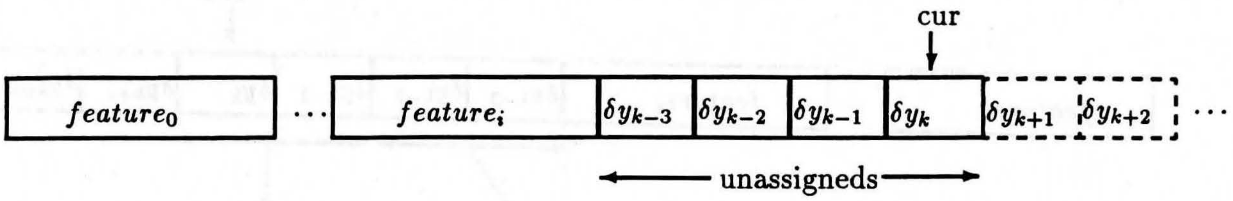

Figure 9: Processing deltas - state 1

When stealing occurs.

The above rules can now be used to state the circumstances under which the various types of feature can steal.

- *Stealing by inclines or declines* : An incline or decline can steal a delta if both Rule G1 and Rule MSC are obeyed for the robber, Rule G1 is obeyed for the victim, and, if the victim is an incline or decline, Rule MSC is obeyed for the victim.
- *Stealing by flats* : A flat can steal a delta from a flat or from an insignificant if Rule G1 is obeyed for both the robber and the victim. A flat cannot steal from an incline or decline.
- *Stealing by insignificants* : Insignificants cannot steal.

Sketch of the algorithm.

Figure 9 shows a general state of processing. Deltas are represented as small boxes and features are represented by rectangles. Time (the independent variable of the signal) increases from left to right. The last delta that has been read in the input is labeled *cur*. *Cur* has not yet been used in building a feature. To the immediate left of *cur* are several deltas that have been processed but have also not yet been assigned to a feature. To the left of these deltas is the sequence of features $feature_0, \cdots, feature_i$ that have already been formed. Each of these features incorporate one or more deltas. To the right of *cur* are the deltas that have not yet been read by the algorithm. In general the algorithm processes the sequence of deltas starting with the earliest and moving toward the latest. However, at times, the algorithm will go back and reprocess deltas (reassign them via stealing) and, in so doing, modify some of the previously defined features.

Referring to Figure 9, let us assume that the algorithm has just read the delta labeled *cur*. *Cur* thus joins the set of unassigned deltas. Starting from this general state of affairs, we describe the workings of the algorithm. It may be helpful to periodically refer to Figure 11 which shows the flow of the algorithm.

First the algorithm processes the unassigned deltas from right to left looking for a suffix sequence that constitutes a significant feature. It first tries to identify *cur* by itself as a

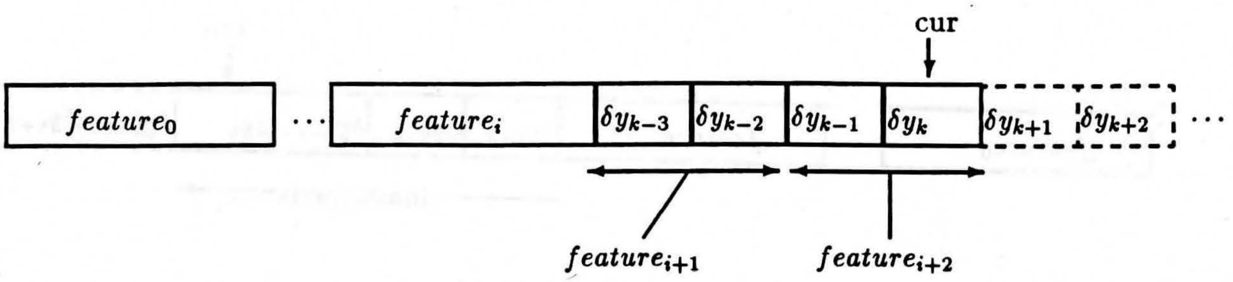

Figure 10: Processing deltas - state 2.

significant feature. If this fails, it tries to identify the suffix $[\Delta y_{k-1}, cur]$ as a significant feature. This continues until a feature is found, or until all the unassigneds have been processed without success, at which time the next delta, Δy_{k+1}, is read and the process repeats.

When a suffix sequence is identified as a new significant feature, in general there will still be some remaining unassigned deltas to its left. These deltas are immediately formed into an insignificant feature. For purposes of illustration let us assume that Δy_{k-1} and Δy_k were found to form a significant feature and that Δy_{k-2} and Δy_{k-3} were formed into an insignificant feature. This is shown in Figure 10.

The algorithm next tries to expand the newly found significant $feature_{i+2}$ to the left as far as possible. First it will try to steal deltas from the newly formed insignificant $feature_{i+1}$. There are two cases :

Case 1 $Feature_{i+2}$ is successful in stealing all of the deltas from $feature_{i+1}$:
$Feature_{i+2}$ then tries to steal deltas from the next feature to the left, $feature_i$. This process continues until it encounters a delta that it cannot steal. Any feature that has all of its deltas stolen goes out of existence. Any feature that has only some of its deltas stolen continues to survive as a feature of the same type, albeit a narrower, modified one.

Case 2 $Feature_{i+2}$ does not steal all of the deltas from $feature_{i+1}$:
The algorithm now lets the previous significant $feature_i$ try to expand to the right by stealing from the remaining deltas in the insignificant $feature_{i+1}$. It is at this point in the algorithm that any feature gets to expand to the right. If $feature_i$ successfully expands to the right, then $feature_i$ is allowed once again to try to expand to the left. This is because, after expanding to the right, $feature_i$ may have changed in a way that now enables it to steal deltas on the left that it couldn't steal in the past. For example, by expanding right, $feature_i$'s slope may have increased so that adding a delta on its left now lets the feature pass a slope test that it failed before.

After both of these case paths, the algorithm loops by resetting cur to the next un-read delta, δy_{k+1}. Upon running out of deltas to read, any remaining unassigned deltas are formed into a terminating insignificant feature, and the prior feature is given the opportunity to steal on its right from this insignificant. Finally, the features are printed out from earliest to last.

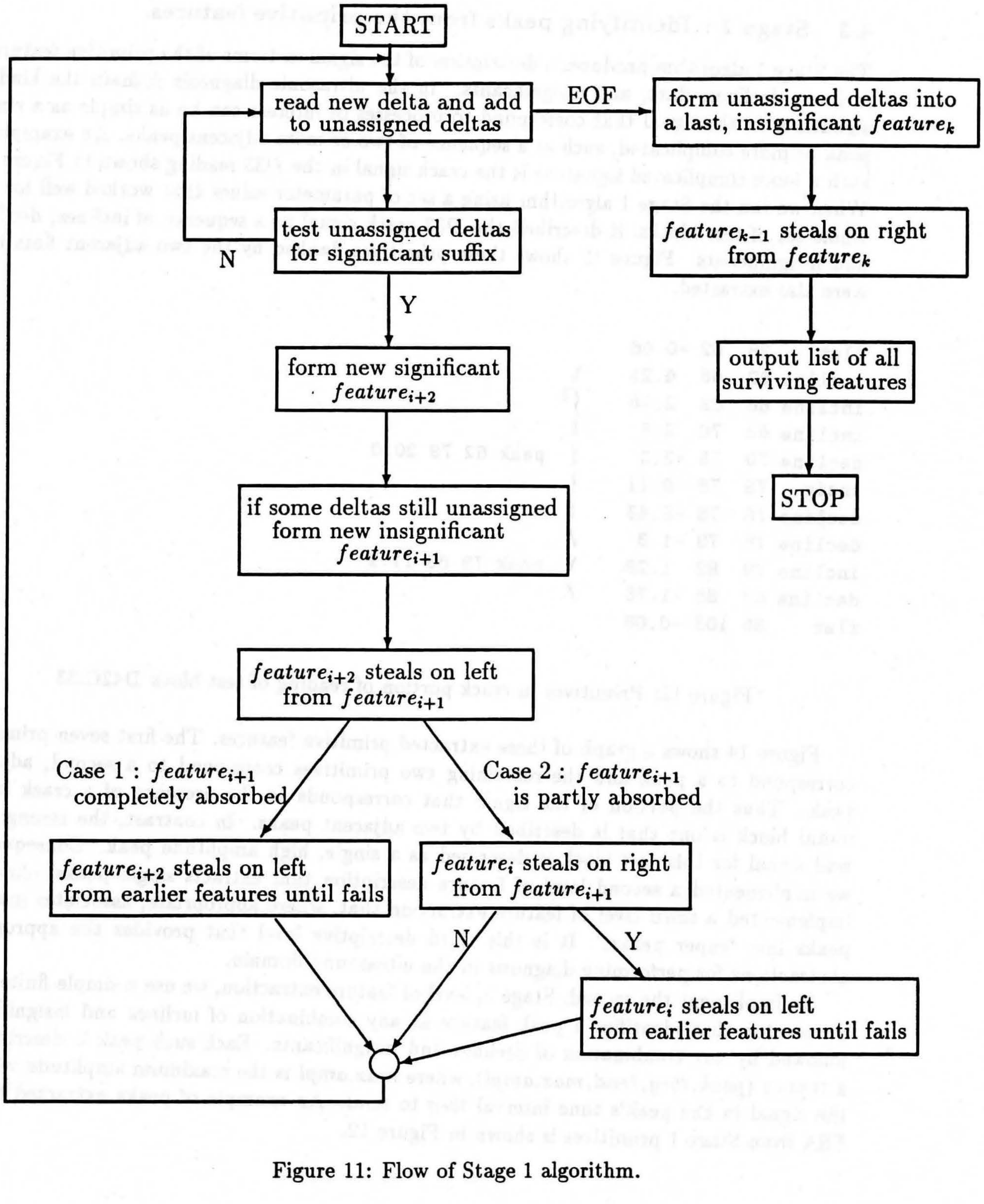
START
read new delta and add to unassigned deltas
EOF
form unassigned deltas into a last, insignificant $feature_k$
test unassigned deltas for significant suffix
N
Y
$feature_{k-1}$ steals on right from $feature_k$
form new significant $feature_{i+2}$
output list of all surviving features
STOP
if some deltas still unassigned form new insignificant $feature_{i+1}$
$feature_{i+2}$ steals on left from $feature_{i+1}$
Case 1 : $feature_{i+1}$ completely absorbed
Case 2 : $feature_{i+1}$ is partly absorbed
$feature_{i+2}$ steals on left from earlier features until fails
$feature_i$ steals on right from $feature_{i+1}$
N
Y
$feature_i$ steals on left from earlier features until fails

Figure 11: Flow of Stage 1 algorithm.

4.3 Stage 2 : Identifying peaks from the primitive features.

The Stage 1 algorithm produces a description of the signal in terms of the primitive features : inclines, declines, flats, and insignificants. In the ultrasonic diagnosis domain the kind of signatures in the signal that correspond to processes of interest can be as simple as a single peak or more complicated, such as a sequence of two or more adjacent peaks. An example of such a more complicated signature is the crack signal in the *D*33 reading shown in Figure 13. When we ran the Stage 1 algorithm using a set of parameter values that worked well for the whole set of test blocks, it described the *D*33 crack signal as a sequence of inclines, declines and insignificants. Figure 12 shows these primitives flanked by the two adjacent flats that were also extracted.

```
flat    28  62 -0.66
incline 62  66  4.21    \
incline 66  69  2.45    |
incline 69  70  2.5     |
decline 70  75 -3.3     |  peak 62 79 20.0
insig   75  76  0.11    |
decline 76  78 -5.43    |
decline 78  79 -1.3     /
incline 79  82  1.29    \  peak 79 85 11.2
decline 82  85 -1.75    /
flat    85 103 -0.08
```

Figure 12: Primitives in crack portion of reading of test block D42C.33

Figure 14 shows a graph of these extracted primitive features. The first seven primitives correspond to a peak and the remaining two primitives correspond to a second, adjacent peak. Thus the portion of the signal that corresponds to the presence of a crack in the metal block is one that is described by two adjacent peaks. In contrast, the strong front wall signal for this test block is described as a single, high amplitude peak. Consequently, we implemented a second level of feature description that extracts single peaks. And, we implemented a third level of feature extraction that, where appropriate, associates multiple peaks into 'super peaks.' It is this third descriptive level that provides the appropriate abstractions for performing diagnosis in the ultrasound domain.

To implement the second, Stage 2, level of feature extraction, we use a simple finite state automaton that identifies a peak feature as any combination of inclines and insignificants followed by any combination of declines and insignificants. Each such peak is described as a tuple : $(peak, tbeg, tend, max.ampl)$, where $max.ampl$ is the maximum amplitude value of the signal in the peak's time interval *tbeg* to *tend*. An example of peaks extracted by the FSA from Stage 1 primitives is shown in Figure 12.

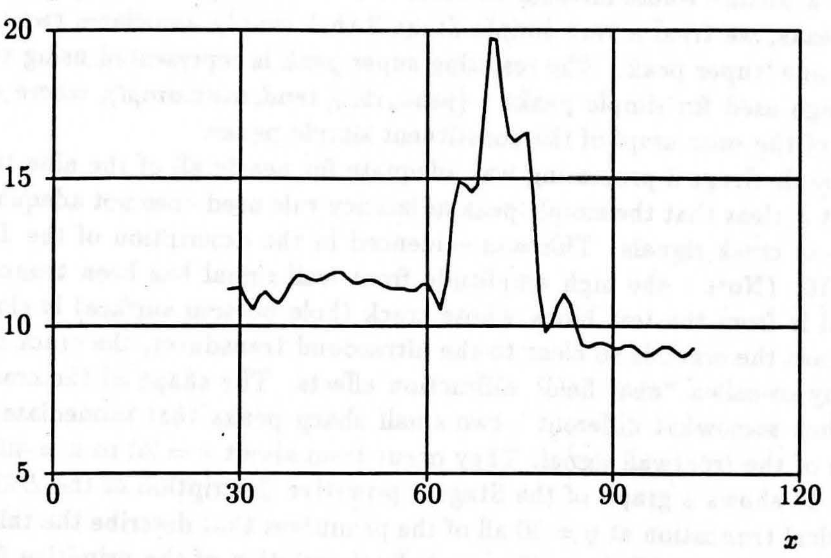

Figure 13: Crack portion of reading of test block D42C.33

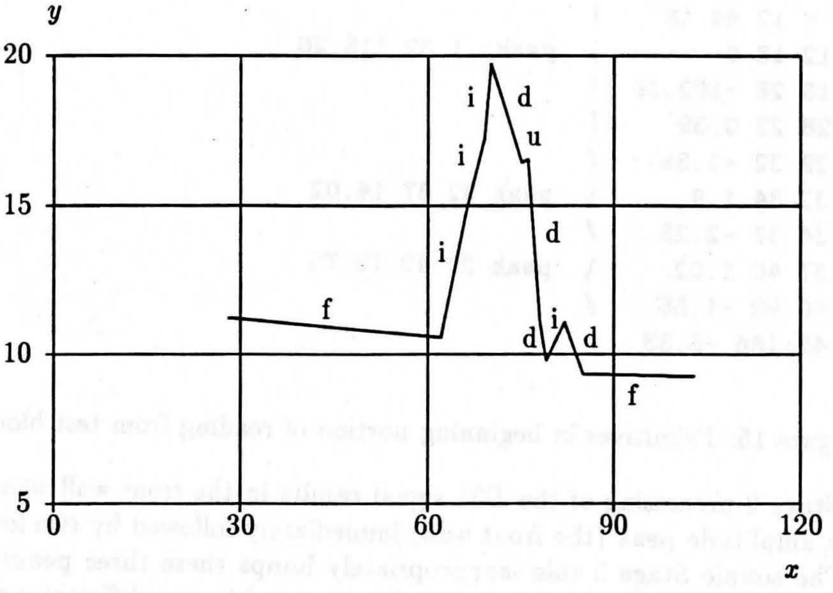

Figure 14: Primitive features for crack of test block D42C.33

4.4 Stage 3 : Identifying 'super peaks' from the peak features.

To identify a feature whose internal structure results in Stage 2 describing it as a sequence of adjacent peaks, we tried a very simple Stage 3 that simply associates two or more adjacent peaks into one 'super peak.' The resulting super peak is represented using the same description language used for simple peaks : (*peak*, *tbeg*, *tend*, *max.ampl*), where *max.ampl* is the maximum of the *max.ampl* of the constituent simple peaks.

This simple Stage 3 processing was adequate for nearly all of the nine test block signals. However, it is clear that the simple peak adjacency rule used does not adequately characterize the nature of crack signals. This was evidenced in the description of the *D*31 signal shown in Figure 16. (Note : the high amplitude front wall signal has been truncated at $y = 30$.) This signal is from the test block whose crack (hole bottom surface) is closest to the front wall. Because the crack is so close to the ultrasound transducer, the crack signal obtained is modified by so-called "near field" diffraction effects. The shape of the crack signal for this block is thus somewhat different - two small sharp peaks that immediate follow the steep downslope of the frontwall signal. They occur from about $x = 30$ to $x = 40$.

Figure 17 shows a graph of the Stage 1 primitive description of the *D*31 signal. Because of the vertical truncation at $y = 30$ all of the primitives that describe the tall front wall signal are not shown. Figure 15 shows the symbolic description of the primitive features extracted for the beginning portion of the *D*31 signal. Also shown in Figure 15 are the peaks extracted by Stage 2.

```
incline  1 6  67.08     \
incline  6 12 44.18     |
insig   12 13 0         |  peak  1 32 115.26
decline 13 28 -102.18   |
insig   28 29 0.39      |
decline 29 32 -1.35     /
incline 32 34 1.9       \  peak 32 37 14.02
decline 34 37 -2.25     /
incline 37 40 1.02      \  peak 37 42 12.79
decline 40 42 -1.55     /
flat    42 166 -5.33
```

Figure 15: Primitives in beginning portion of reading from test block D42C.31

The Stage 2 processing of the *D*31 signal results in the front wall signal being described by a high amplitude peak (the front wall) immediately followed by two low amplitude crack peaks. The simple Stage 3 rule inappropriately lumps these three peaks together into one super peak. This situation might possibly be corrected in two different ways :

- One could try to implement a more complicated Stage 3 that uses knowledge about the shape of peaks and their distribution in the signal. For example, from prior observations

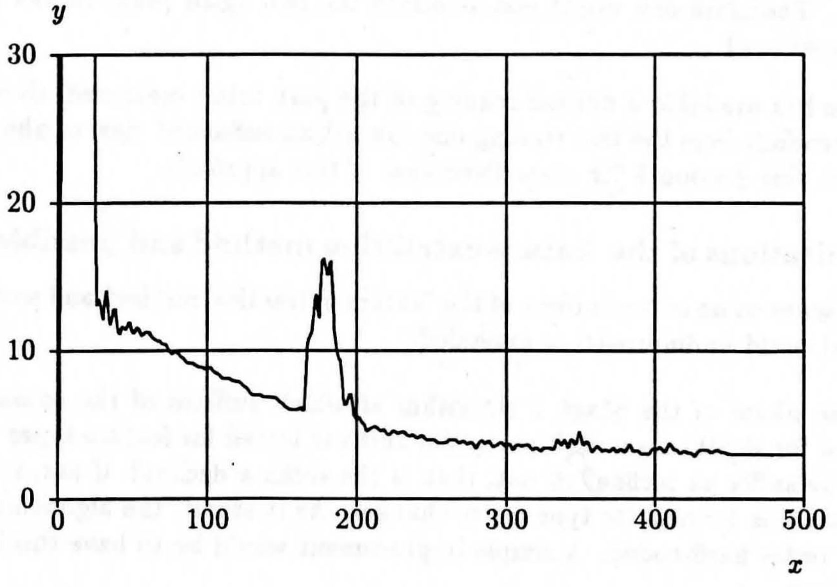

Figure 16: Reading of test block D42C.31

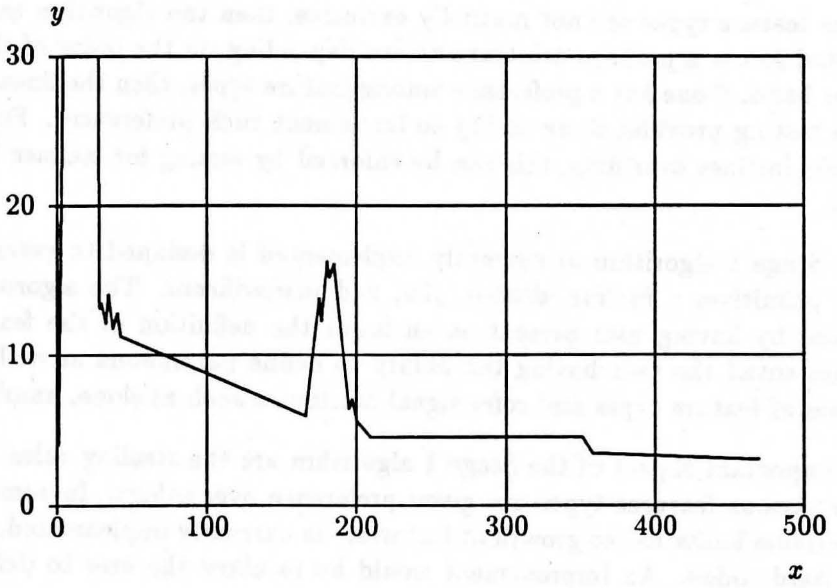

Figure 17: Primitive features extracted for D31 signal.

one might know that the front wall signal is normally a simple, clean large amplitude peak. Therefore one would not associate the two small peaks in the $D31$ signal with the front wall.

- If one has available a normal reading of the part being measured, then by subtracting this reading from the test reading one can get an enhanced view of abnormalities in the signal. See Section 5 for more discussion of this approach.

4.5 Limitations of the feature extraction method and possible future work.

Here we discuss some of limitations of the feature extraction method and some ways in which the method could be improved or extended.

- In the phase of the Stage 1 algorithm at which suffixes of the unassigned deltas are tested for significance, each particular suffix is tested for feature types in a fixed order : "Is the suffix an incline? If not, then is the suffix a decline? If not, is it a flat?" Once a feature is formed, its type never changes. As it stands the algorithm has the feature type order hard-coded. A simple improvement would be to have this order defined via an input.

  The fixed ordering of testing for feature type introduces no inaccuracy of classification so long as the feature types are mutually exclusive. That is, if a given delta sequence cannot be classified as a significant feature in more than one way. This orthogonality can be achieved by judiciously choosing the numeric input parameters.

  If the feature types are not mutually exclusive, then the algorithm as currently implemented forces a perhaps arbitrary choice depending on the order of the types. On the other hand, if one has a preference among feature types, then the linear order of feature type testing provides some ability to implement such preferences. For example, if one prefers inclines over flats, this can be enforced by testing for inclines before testing for flats.

- The Stage 1 algorithm as currently implemented is designed to extract effectively the four primitives : *incline*, *decline*, *flat*, and *insignificant*. The algorithm could be improved by having user present as an input the definition of the feature types. This would entail the user having the ability to define parameters as well as introduce the names of feature types and refer signal attributes such as slope, amplitude, width, etc.

- An important aspect of the Stage 1 algorithm are the stealing rules. These determine how various features types are given preference over others. In some cases, they also determine limits to the growth of features. As currently implemented, the stealing rules are hard-coded. An improvement would be to allow the user to define these stealing rules as an input to algorithm. This improvement and the one mentioned immediately above would involve developing a system that applies a user-defined rulebase under the control of an algorithm identical or similar to that shown in Figure 11.

- A possible difficulty with our feature extraction method is that five numeric parameters need to be set by the user. While these parameters ostensibly have a simple semantics, the following potential issues might arise when using the method :
  1. In specific instances, the competitive process by which the Stage 1 algorithm extracts adjacent features might result in some parsings of signal segments that look somewhat unnatural or suboptimal to the human eye.

     For example, in Figure 17 the signal segment from $x = 210$ to $x = 470$ is described by two flats with an intervening small decline. A human might well describe the whole segment as one flat. If the value of the parameter *sig.ampl.change* is increased a small amount, for example, from 1 to 1.2, then the part of the segment that was previously extracted as a decline no longer so-qualifies because the amplitude decrease is now too small, and the segment then gets described as one large flat.
  2. The parameter change described above (from 1 to 1.2) results in other small but significant changes in the primitive description of the $D31$ signal. One of these changes is that the crack portion of the $D31$ reading, which used to be described by

     ```
     incline 32 34 1.9
     decline 34 37 -2.25
     incline 37 40 1.02
     decline 40 42 -1.55
     ```

     is now described by

     ```
     incline 32 34 1.9
     decline 34 37 -2.25
     insig 37 40 1.02
     decline 40 42 -1.55
     ```

     Accordingly, what Stage 2 previously described in the interval $x = 32$ to $x = 42$ as a sequence of two peaks, now is described as one peak.

  Given numeric parameters, there is an inevitable thresholding sensitivity to their exact values. This suggests that what is needed to obtain a robust extraction of features at the more abstract, diagnostic level is Stage 3 processing in which the features from Stage 2 (or perhaps the features directly from Stage 1) are used to focus attention on signal segments, to which is then applied knowledge-based inference that can recognize multiple differing descriptions (such the two given above) as indicating an identical underlying diagnostic cause (such as a crack in the range $x = 32$ to $x = 42$).
- An interesting extension to this research would be to develop a system that tries to find parameter settings that can reproduce (as much as possible) a given expert's examples of feature extraction. Such a supervised-learning system might provide a useful and efficient user interface for using the feature extraction method.

Table 1: Features extracted for the nine test blocks.

$$\begin{aligned} \textit{Parameter settings}: \quad & sig.ampl.change = 1.0 \\ & sig.slope = 0.1 \\ & sig.flat.width = 10.0 \\ & sig.flat.slope = 0.05 \\ & max.slope.change = 0.7 \end{aligned}$$

	peak #1			peak #2			peak #3			peak #4		
Reading	*tbeg*	*tend*	*max. ampl*	*tbeg*	*tend*	*max. ampl*	*tbeg*	*tend*	*max. ampl*	*tbeg*	*tend*	*max. ampl*
D31	1	42	115.26	166	209	16.23						
D32	1	32	115.26	37	54	15.97	179	206	17.21			
D33	1	28	115.26	62	85	19.71	205	233	14.29	418	455	6.65
D34	1	32	115.16	85	107	18.64	230	265	15.25			
D35	1	32	115.26	110	132	14.33	255	290	18.93			
D36	1	32	115.16	136	151	11.07	281	308	17.94			
D37	2	32	115.12	159	175	10.09	291	340	20.54			
D38	1	36	114.12	188	204	6.69	319	358	16.41			
D39	1	32	115.32	211	227	6.42	346	384	18.90			

4.6 Performance of the feature extraction method.

Using a common set of numerical parameters across all nine test blocks, the feature extraction method was able to extract the crack feature from eight of the nine test blocks. Table 1 shows the final feature descriptions produced for nine test block readings using the indicated parameter settings.

Only two peaks were extracted from the *D*31 reading because the peaks associated with the crack signal were incorrectly merged with the front wall signal to form one super peak. This was due primarily to the simplicity of the Stage 3 feature processing in which two or more peaks were merged into super peaks. In this signal, the very large front wall peak is followed by the crack signal described as two very small peaks. A more sophisticated heuristic in which prior knowledge about relative peak size and placement guides the merging process could probably avoid such a confoundment. In other words, knowledge about the size, shape and structure of various kinds of peaks could be brought to bear in analyzing the simple peak features extracted by the Stage 2 of processing.

The *D*33 reading contains four peaks. The fourth peak is an "echo" resulting from part of the back wall echo reflecting off the front of the block, then traveling to the back wall and producing another back wall reflection.

5 Feature extraction from differenced test block signals

Figure 18 is identical to Figure 3. It graphs an ultrasonic reading from a $D38$ test block which shows a 'crack' (hole reflection). This is one of the more difficult test block readings. It is difficult because the crack reflection at $x = 200$ is small and is similar in size to the little "hillock" that occurs in all the readings from about $x = 30$ to $x = 60$. This hillock does not indicate any kind of defect and is a normal part of a test block reading. Any algorithm that works by identifying peaks by keying on amplitude changes, and that attempts to extract the crack features from all nine of the test block using a common set of parameters, will be strained to succeed when the cracks become as small as normal features. There will be a tendency to extract the hillock feature as well as the crack feature.

As mentioned in Section 3.1 one approach to handling this difficulty is to subtract from each test reading a prototypical normal reading, and then extract features from this differenced signal. Ten readings from a location of the D38 block that gives no crack signal (ie., ten normal readings) were pointwise-averaged together to obtain a 'normal' reading for the block. This normal reading is subtracted from the reading to be analyzed. Figure 19 shows the differenced version of the signal in Figure 18. The resulting 'differenced' signal is one in which the front wall, the hillock and the gentle exponential-like decay following the front wall peak, essentially disappear. What survives is a flat signal around which low amplitude noise oscillates. Also surviving prominently is the crack signal and the back wall signal. The back wall survives as a 'zig zag' signal. This is because the back wall peak in a reading of a 'cracked' test block is reduced in amplitude and also shifted forward in a little time.

By subtracting a normal reading of the part from a test reading, abnormal features are enhanced in their amplitudes relative to the noise that remains. They are therefore easier to detect by feature extraction algorithms that key on amplitude changes.

6 Conclusions and recommendations.

The foregoing describes some experience we have had with two approaches to feature extraction : one based upon techniques used in the Scale-space method, and the other being a feature extracted algorithm developed at UC Irvine. Some other techniques used in feature extraction were also briefly discussed. We next compare the two feature extraction methods and make some recommendations for further testing and development.

6.1 Scale-space techniques.

With regard to Scale-space techniques, we conclude that for the task of extracting features from signals, the full-blown Scale-space method, with it generation of an interval tree describing the signal at multiple scales, is computational overkill. Using a Scale-space approach (multiple filterings, generation and following of contours) requires considerable computation. We found that Scale-space contour generation produced contours whose strength did not always correlate with the feature strength as measured from an amplitude perspective.

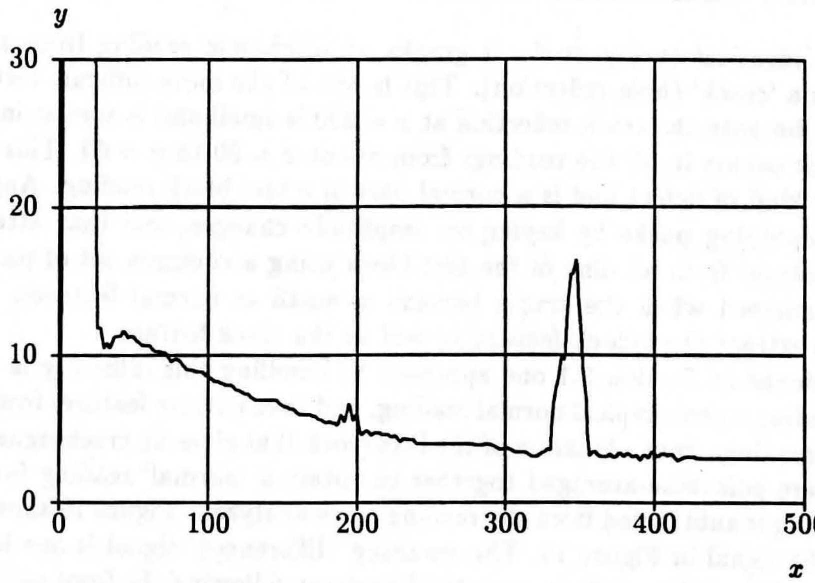

Figure 18: Reading of test block D42C.38

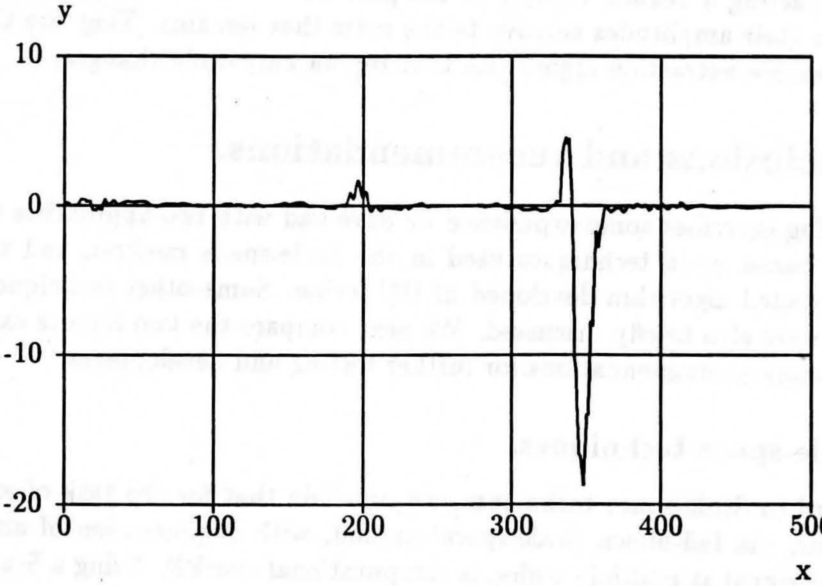

Figure 19: Differenced signal for test block D42C.38

At the expense of more computation, a Filtered Scale-space approach that uses a post-processing filtering phase was tested. This approach helped eliminate strong contours associated with weak features, but this process also tended to obliterate some of the contours of the features of interest. We were not able to extract all the crack features from our test suite of nine test block readings using this method.

6.2 Our feature extraction method.

We developed and tested a new feature extraction algorithm as described in Section 4. This method was found to be more capable than Filtered Scale-space at extracting the (small) crack features in the test suite of nine test block readings. It was also found, in practice, to be computationally efficient compared to Filtered Scale-space.

With respect to comparative crack feature extraction capability, Filtered Scale-space was unable to extract the crack features for the readings $D31$ and $D38$. Our feature extraction method succeeded with $D38$. And with the implementation of a Stage 3 phase that uses some domain knowledge about feature geometries and locations, this method should be able to extract the crack feature in the $D31$ reading.

With respect to comparative processing speed, on a Sun 3/60 with 16 megabytes of memory, Filtered Scale-space took on the order of tens of minutes to extract features from one reading using a scale space based on 92 filterings. In contrast, our feature extraction algorithm, on the same platform, extracted features from one reading in time on the order of a few seconds. Admittedly, the Filtered Scale-space approach could be speeded-up by determining a smaller set of filters that are sufficient to still capture useable contours - say 10 filters instead of 92. The C code in which Filtered Scale-space is implemented might also be optimized somewhat. However it is difficult to see how these changes could result in a processing time that approaches that of a few seconds.

Our feature extraction method does not require any smoothing of the signal being analyzed. It can operate simply by processing the raw signal described by the time series of amplitude changes between the sampling points - the so-called *deltas*. On the other hand, the method could just as well be given a smoothed signal to process. Such a signal would in general be describable in terms of fewer deltas. [2] This would result in Stage 1 of the method having fewer deltas to process and thus generally running faster. Whether any significant speed-up could be obtained by first smoothing and then doing feature extraction has not been tested. Such an approach would also have to consider any trade-offs involved in lost signal detail as a result of the smoothing and the possibility of losing significant diagnostic information.

Our feature extraction method works well on the signals in the ulrasonic test block domain. To determine whether this method works well on other signals, and to determine how easy the method is to use vis à vis setting parameters, we recommend that the method be field-tested on a variety of signals by users unfamiliar with the internal algorithmic details of the method.

[2] Alternatively, one could preprocess the signal by describing each monotonic signal segment by one delta, at the possible risk of losing significant signal 'shape' information.

References

Amirfathi, M. M., Morris, S., O'Rorke, P., Bond, W. E., & St. Clair, D. C. (1991). Pattern Recognition for Non-Destructive Evaluation. In L. Mallette (Ed.), *Proceedings of the 1991 IEEE Aerospace Applications Conference.* Crested Butte, CO: IEEE Computer Society Press.

Chen, C. H. (1982) Introduction. In C. H. Chen (Ed.), *Digital waveform processing and recognition* (pp. 1-3).

Gallagher, J. P., Giessler, F. J., Berens, A. P., & Engle, R. M. (1984). *USAF Damage Tolerant Design Handbook: Guidelines for the Analysis and Design of Damage Tolerant Aircraft Structures,* (Technical Report Air Force Report No. AFWAL-TR-82-3073). Wright-Patterson Air Force Base.

Love, P. L., & Simaan, M. (1988). Automatic recognition of primitive changes in manufacturing process signals. *Pattern Recognition, 21,* 333-342.

Marr, D., & Hildreth, E. (1980) Theory of edge detection. *Proceedings of the Royal Society of London, B 207,* 187-217.

O'Rorke, P., Morris, S., Amirfathi, M., Bond, W. E., & St. Clair, D. C. (1991). Machine learning for nondestructive evaluation. In *Proceedings of the Eighth International Workshop on Machine Learning.*

Ullman, S. (1986). Artificial intelligence and the brain : Computational studies of the visual system. *Annual Review of Neuroscience, 9,* 1–26.

Witkin, A. P. (1983). Scale-space filtering. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (pp. 1019-1022). Karlsruhe, West Germany: Morgan Kaufmann.