TECHNICAL RESEARCH REPORT

Fast Digital Locally Monotonic Regression

by N.D. Sidiropoulos

T.R. 95-80

ISR
INSTITUTE FOR SYSTEMS RESEARCH

Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

Fast Digital Locally Monotonic Regression

N.D. Sidiropoulos

Abstract

In [1], Restrepo and Bovik developed an elegant mathematical framework in which they studied locally monotonic regressions in $\mathbf{R}^N$. The drawback is that the complexity of their algorithms is exponential in N. In this paper, we consider *digital* locally monotonic regressions, in which the output symbols are drawn from a finite alphabet, and, by making a connection to Viterbi decoding, provide a fast $O(|\mathcal{A}|^2 \alpha N)$ algorithm that computes any such regression, where $|\mathcal{A}|$ is the size of the digital output alphabet, α stands for lomo-degree, and N is sample size. This is *linear* in N, and it renders the technique applicable in practice.

Keywords

Nonlinear Filtering, Local Monotonicity, Principle of Optimality, Viterbi Algorithm

EDICS: SP 2.7.3; also, SP 6.1.7

N.D. Sidiropoulos is with the Institute for Systems Research, University of Maryland, College Park, MD 20742 U.S.A. He can be reached at (301) 405-6591, or via e-mail at nikos@Glue.umd.edu

I. Introduction

Local monotonicity is a property that appears in the study of the set of root signals of the median filter [2], [3], [4], [5], [6], [7]; it constraints the roughness of a signal by limiting the rate at which the signal undergoes changes of trend (increasing to decreasing or vice versa). In effect, it *limits the frequency of oscillations, without limiting the magnitude of jump level changes that the signal exhibits.* Local monotonicity implies a different notion of smoothness, as compared to e.g., limiting the support of the Fourier transform; the latter imposes a limit on *both* the frequency of oscillations, *and* the magnitude of jump level changes.

A classic problem in the true spirit of nonlinear filtering is the recovery of a piecewise smooth signal embedded in impulsive noise. In this paradigm, it is often natural to model the signal as locally monotonic, and ask for optimal smoothing under an approximation or estimation criterion. This amounts to picking a signal, from a given class of locally monotonic signals, which minimizes a distortion measure between itself and the observation, and it is referred to as *locally monotonic regression.* In [1], Restrepo and Bovik developed an elegant mathematical framework in which they studied locally monotonic regressions in $\mathbf{R}^N$ (throughout, $\mathbf{R}$ denotes the set of real numbers, and $|\cdot|$ stands for set cardinality). Unfortunately, the complexity of their algorithms is exponential in N. The authors admit that their algorithms are computationally very expensive, even for signals of relatively short duration; this hampers potential applications of the method.

Locally monotonic regression provides a median root which is optimal in a suitable sense, e.g., closest to the observable data in some metric or semi-metric. It is meant as an "optimal median", while iterating the median may be thought of as a suboptimal "regression" which trades optimality for simplicity. In practice, one usually deals with digital (finite-alphabet) data. If the input (observable data) is finite-alphabet, then the output of any number of iterations of the median is also finite-alphabet, and, in fact, of the same alphabet as the input; it is therefore natural to consider digital locally monotonic regression, in which the output symbols are drawn from a finite alphabet, as the optimal counterpart of median filtering of digital signals. Even if the observable data is real-valued, one would probably still be interested in digital locally monotonic regression, for, on one hand, by proper choice of quantization, it may provide an answer which is sufficiently close to the underlying regression in $\mathbf{R}^N$, and that may well be all that one cares for; and, on the other hand, it provides a way to perform simultaneous smoothing, quantization, and compression of noisy discontinuous signals. In this paper, we consider digital locally monotonic regression, and, by making a connection to Viterbi decoding[1], provide a fast $O(|\mathcal{A}|^2\alpha N)$ algorithm that computes any such regression, where $|\mathcal{A}|$ is the size of the digital output alphabet, α is the lomo-degree (usually, the assumed lomotonicity of the signal, i.e., the highest degree of local monotonicity that the signal possesses), and N is the size of the sample. This is *linear* (as opposed to *exponential* in the work of Restrepo and Bovik) in N, and it renders the technique applicable in practice.

In more consice terms, we provide a fast $O(|\mathcal{A}|^2\alpha N)$ Viterbi-type algorithm that solves the following problem. Given a sequence of finite extent, $\mathbf{y} = \{y(n)\}_{n=0}^{N-1} \in \mathbf{R}^N$, find a finite-alphabet sequence, $\widehat{\mathbf{x}} = \{\widehat{x}(n)\}_{n=0}^{N-1} \in \mathcal{A}^N$, which minimizes $d(\mathbf{x},\mathbf{y}) = \sum_{n=0}^{N-1} d_n(y(n), x(n))$ subject to: $\mathbf{x}$ is locally monotonic of degree α.

A. Organization

The rest of this paper is structured as follows. In section II we provide some necessary definitions, and a formal statement of the problem. The reader is referred to [1] and references therein for additional background and motivation. Our fast solution is presented in section III. A discussion on implementation complexity is also included. A complete simulation experiment is presented in section IV, and conclusions are drawn in section V.

II. The Problem

A. Background

If $\mathbf{x}$ is a real-valued sequence (string) of length N, and γ is any integer less than or equal to N, then a *segment* of $\mathbf{x}$ of length γ is any substring of γ consecutive components of $\mathbf{x}$. Let $\mathbf{x}_i^{i+\gamma-1} = \{x(i), \cdots, x(i+\gamma-1)\}$, $i \geq 0$, $i+\gamma \leq N$, be any such segment. $\mathbf{x}_i^{i+\gamma-1}$ is monotonic if either $x(i) \leq x(i+1) \leq \cdots \leq x(i+\gamma-1)$, or $x(i) \geq x(i+1) \geq \cdots \geq x(i+\gamma-1)$.

Definition 1: A real-valued sequence, $\mathbf{x}$, of length N, is *locally monotonic* of degree $\alpha \leq N$ (or *lomo-α*, or simply *lomo* in case α is understood) if each and every one of its segments of length α is monotonic.

Throughout the following, we assume that $3 \leq \alpha \leq N$. A sequence $\mathbf{x}$ is said to exhibit an increasing (resp. decreasing) transition at coordinate i if $x(i) < x(i+1)$ (resp. $x(i) > x(i+1)$).

[1]Such a connection between optimal nonlinear filtering under local syntactic constraints and Viterbi decoding algorithms has first been made in [8].

If $\mathbf{x}$ is locally monotonic of degree α, then $\mathbf{x}$ has a constant segment (run of identical symbols) of length at least $\alpha - 1$ in between an increasing and a decreasing transition. The reverse is also true. If $3 \leq \alpha \leq \beta \leq N$, then a sequence of length N that is lomo-β is lomo-α as well; thus, the *lomotonicity* of a sequence is defined as the highest degree of local monotonicity that it possesses.

B. Digital Locally Monotonic Regression

Given $y(n) \in \mathbf{R}$, $n = 0, 1, \cdots, N-1$, and $\mathcal{A}$, a finite subset of $\mathbf{R}$ ($|\mathcal{A}| < \infty$). Let $\Lambda(\alpha, N, \mathcal{A})$ denote the space of all sequences of N elements of $\mathcal{A}$ which are locally monotonic of degree α. Digital locally monotonic regression is the following constrained optimization:

$$\textbf{minimize} \sum_{n=0}^{N-1} d_n(y(n), x(n)) \tag{1}$$

$$\textbf{subject to}: \mathbf{x} = \{x(n)\}_{n=0}^{N-1} \in \Lambda(\alpha, N, \mathcal{A}) \tag{2}$$

Here, $d_n(\cdot, \cdot)$ is any per-letter distortion measure; it can be a - possibly inhomogeneous in n - metric, semi-metric, or arbitrary bounded cost measure. The "sum" may also be interpreted liberally: it turns out that it can be replaced by a "max" operation to accommodate a minimax (minimize sup-error) problem formulation, without affecting the structure of the fast computational algorithm which is developed below.

Observe that if $3 \leq \alpha \leq \beta \leq N$, then $\Lambda(\beta, N, \mathcal{A}) \subseteq \Lambda(\alpha, N, \mathcal{A})$; thus, the above optimization is defined over an element of a sequence of nested "approximation" spaces. This means that the achievable minimum is a non-decreasing function of α.

III. Solution

We show how a suitable reformulation of the problem naturally leads to a simple and efficient Viterbi-type optimal algorithmic solution.

Definition 2: Given any sequence $\mathbf{x} = \{x(n)\}_{n=0}^{N-1}$, $x(n) \in \mathcal{A}$, $n = 0, 1, \cdots, N-1$, define its associated *state sequence*, $\mathbf{s_x} = \left\{[x(n), l_{\mathbf{x}}(n)]^T\right\}_{n=-1}^{N-1}$, where $[x(-1), l_{\mathbf{x}}(-1)]^T = [\phi, \alpha - 1]^T$, $\phi \notin \mathcal{A}$ and, for $n = -1, \cdots, N-2$

$$l_{\mathbf{x}}(n+1) = \begin{cases} sgn(l_{\mathbf{x}}(n)) \cdot min\left\{abs(l_{\mathbf{x}}(n)) + 1,\ \alpha - 1\right\} & , \ x(n+1) = x(n) \\ 1 & , \ x(n+1) > x(n) \\ -1 & , \ x(n+1) < x(n) \end{cases}$$

where $sgn(\cdot)$ stands for the sign function, and $abs(\cdot)$ stands for absolute value. $[x(n), l_{\mathbf{x}}(n)]^T$ is the state at time n, and, for $n = 0, 1, \cdots, N-1$, it assumes values in $\mathcal{A} \times \{-(\alpha-1), \cdots, -1, 1, \cdots, \alpha-1\}$. Clearly, we can equivalently pose the optimization (1),(2) in terms of the associated state sequence.

Definition 3: A subsequence of state variables $\left\{[x(n), l_{\mathbf{x}}(n)]^T\right\}_{n=-1}^{\nu}$, $\nu \leq N-1$, is *admissible* (with respect to constraint (2)) if and only if there exists a suffix string of state variables, $\left\{[x(n), l_{\mathbf{x}}(n)]^T\right\}_{n=\nu+1}^{N-1}$, such that $\left\{[x(n), l_{\mathbf{x}}(n)]^T\right\}_{n=-1}^{\nu}$ followed by $\left\{[x(n), l_{\mathbf{x}}(n)]^T\right\}_{n=\nu+1}^{N-1}$ is the associated state sequence of some sequence in $\Lambda(\alpha, N, \mathcal{A})$.

Let $\widehat{\mathbf{x}} = \{\widehat{x}(n)\}_{n=0}^{N-1}$ be *a* solution (one always exists, although it may not necessarily be unique) of (1),(2), and $\left\{[\widehat{x}(n), l_{\widehat{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{N-1}$, be its associated state sequence. Clearly, $\left\{[\widehat{x}(n), l_{\widehat{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{N-1}$ is admissible, and so is any subsequence $\left\{[\widehat{x}(n), l_{\widehat{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{\nu}$, $\nu \leq N-1$. The following is a key observation.

Claim 1: Optimality of $\left\{[\widehat{x}(n), l_{\widehat{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{N-1}$ implies optimality of $\left\{[\widehat{x}(n), l_{\widehat{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{\nu}$, $\nu \leq N-1$, among all admissible subsequences of the same length which lead to the same state at time ν, i.e., all admissible $\left\{[\widetilde{x}(n), l_{\widetilde{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{\nu}$ satisfying $[\widetilde{x}(\nu), l_{\widetilde{\mathbf{x}}}(\nu)]^T = [\widehat{x}(\nu), l_{\widehat{\mathbf{x}}}(\nu)]^T$

Proof: The argument goes as follows. Suppose that $\left\{[\widetilde{x}(n), l_{\widetilde{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{\nu}$ is an admissible subsequence of states satisfying $[\widetilde{x}(\nu), l_{\widetilde{\mathbf{x}}}(\nu)]^T = [\widehat{x}(\nu), l_{\widehat{\mathbf{x}}}(\nu)]^T$. It is easy to see that $\left\{[\widetilde{x}(n), l_{\widetilde{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{\nu}$ followed by $\left\{[\widehat{x}(n), l_{\widehat{\mathbf{x}}}(n)]^T\right\}_{n=\nu+1}^{N-1}$ is also admissible. The key point is that any suffix string of state variables which makes $\left\{[\widehat{x}(n), l_{\widehat{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{\nu}$ admissible, will also make $\left\{[\widetilde{x}(n), l_{\widetilde{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{\nu}$ admissible. If $\left\{[\widetilde{x}(n), l_{\widetilde{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{\nu}$ has a smaller cost (distortion) than $\left\{[\widehat{x}(n), l_{\widehat{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{\nu}$, then, by virtue of the fact that the cost is a sum of per-letter costs, $\left\{[\widetilde{x}(n), l_{\widetilde{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{\nu}$ followed by $\left\{[\widehat{x}(n), l_{\widehat{\mathbf{x}}}(n)]^T\right\}_{n=\nu+1}^{N-1}$ will have a smaller cost than $\left\{[\widehat{x}(n), l_{\widehat{\mathbf{x}}}(n)]^T\right\}_{n=-1}^{N-1}$, and this violates the optimality of the latter. ■

This is a particular instance of the principle of optimality of dynamic programming [9], [10], [11]. The following is an important Corollary.

Corollary 1: An optimal admissible path to any given state at time $n+1$ must be an admissible one-step continuation of an optimal admissible path to *some* state at time n.

This Corollary leads to an efficient Viterbi-type [12], [13], [14] algorithmic implementation of any digital locally monotonic regression. It remains to specify the costs associated with one-step state transitions in a way that forces one-step optimality and admissibility. This is not very difficult. Let $c\left(\mathbf{s}_{\mathbf{x}}(n) \rightarrow \mathbf{s}_{\mathbf{x}}(n+1)\right)$ denote the cost of a one-step state transition, and $\vee, \wedge$ denote logical

OR, AND, respectively. Then,

if :

$$(l_{\mathbf{x}}(n+1)=1)\ \wedge\ (x(n)<x(n+1))\ \wedge\ [(l_{\mathbf{x}}(n)>0)\ \vee\ (l_{\mathbf{x}}(n)=-(\alpha-1))]$$

/* To make an increasing transition, one of two things must hold: **either** you're currently in the midst of an increasing trend, **or**, if in the midst of a decreasing trend, you've just completed a constant run of at least $\alpha-1$ symbols following the latest decreasing transition. */

$$\vee$$

$$(l_{\mathbf{x}}(n+1)=-1)\ \wedge\ (x(n)>x(n+1))\ \wedge\ [(l_{\mathbf{x}}(n)<0)\ \vee\ (l_{\mathbf{x}}(n)=\alpha-1)]$$

/* Similarly, to make a decreasing transition, one of two things must hold: **either** you're currently in the midst of a decreasing trend, **or**, if in the midst of an increasing trend, you've just completed a constant run of at least $\alpha-1$ symbols following the latest increasing transition. */

$$\vee$$

$$(1<l_{\mathbf{x}}(n+1)<\alpha-1)\ \wedge\ (x(n)=x(n+1))\ \wedge\ (l_{\mathbf{x}}(n+1)=l_{\mathbf{x}}(n)+1)$$

/* If you are in a constant run following an increasing transition, and you receive one more identical symbol, then the only thing you are allowed to do is increment your counter */

$$\vee$$

$$(-(\alpha-1)<l_{\mathbf{x}}(n+1)<-1)\ \wedge\ (x(n)=x(n+1))\ \wedge\ (l_{\mathbf{x}}(n+1)=l_{\mathbf{x}}(n)-1)$$

/* Similarly, if you are in a constant run following a decreasing transition, and you receive one more identical symbol, then the only thing you are allowed to do is decrement your counter */

$$\vee$$

$$(l_{\mathbf{x}}(n+1)=\alpha-1)\ \wedge\ (x(n)=x(n+1))\ \wedge\ [(l_{\mathbf{x}}(n)=\alpha-1)\ \vee\ (l_{\mathbf{x}}(n)=(\alpha-1)-1)]$$

/* The only way you can reach a positive full count of $\alpha-1$ is to either have a positive full count, or be just one sample short of a positive full count **and** receive one more identical symbol */

$$\vee$$

$$(l_{\mathbf{x}}(n+1)=-(\alpha-1))\ \wedge\ (x(n)=x(n+1))\ \wedge\ [(l_{\mathbf{x}}(n)=-(\alpha-1))\ \vee\ (l_{\mathbf{x}}(n)=-(\alpha-1)+1)]$$

/* The only way you can reach a negative full count of $-(\alpha-1)$ is to either have a negative full count, or be just one sample short of a negative full count **and** receive one more identical symbol */

$$\begin{aligned} \textbf{then}: c\left([x(n), l_{\mathbf{x}}(n)]^T \to [x(n+1), l_{\mathbf{x}}(n+1)]^T\right) &= d_{n+1}(y(n+1), x(n+1)) \\ \textbf{else}: c\left([x(n), l_{\mathbf{x}}(n)]^T \to [x(n+1), l_{\mathbf{x}}(n+1)]^T\right) &= \infty \end{aligned} \tag{3}$$

will do it. A formal proof can be easily constructed, and is hereby omitted.

A. Complexity

Any Viterbi-type algorithm has computational complexity which is *linear* in the number of observations, i.e., N. The number of computations per observation symbol depends on the number of states, as well as state connectivity in the trellis. In the following, we derive the required number of distance (branch metric) calculations and additions per observation symbol (trellis stage) (the number of comparisons required per trellis stage is always less than this number). Each stage in the trellis has a total of $|\mathcal{A}|2(\alpha-1)$ states, which can be classified as follows:

- $|\mathcal{A}|$ state pairs of the form $([v,-1]^T, [v,1]^T)$, $v \in \mathcal{A}$. One can easily check that the combined fun-in of each such pair (i.e., the number of states at the previous time instant from which such a pair can be reached) is $(|\mathcal{A}|-1)\alpha$. Thus, one needs $(|\mathcal{A}|-1)\alpha$ distance calculations and additions per pair, for a subtotal of $|\mathcal{A}|(|\mathcal{A}|-1)\alpha$ distance calculations and additions per stage, for this class of states.
- $|\mathcal{A}|2(\alpha-3)$ states of the form $[v,l]^T$, $v \in \mathcal{A}$, $1 < l < \alpha-1$, or $-(\alpha-1) < l < -1$. Each such state can only be reached by one state, namely $[v, l-1]^T$ if $l > 0$, or $[v, l+1]^T$ otherwise. Thus, one needs $|\mathcal{A}|2(\alpha-3)$ distance calculations and additions per stage, for this class of states.
- $|\mathcal{A}|$ state pairs of the form $([v,-(\alpha-1)]^T, [v,\alpha-1]^T)$, $v \in \mathcal{A}$. One can easily check that the combined fun-in of each such pair is 4. Indeed, a state of type $[v,\alpha-1]^T$ can only be reached from either itself or $[v,(\alpha-1)-1]^T$, and, similarly, a state of type $[v,-(\alpha-1)]^T$ can only be reached from either itself or $[v,-(\alpha-1)+1]^T$. Therefore, one needs $4|\mathcal{A}|$ distance calculations and additions per stage, for this class of states.

The total is $|\mathcal{A}|^2\alpha + |\mathcal{A}|(\alpha-2)$ distance calculations and additions per stage; this is tabulated in Table I, for some typical parameter values, and it is of $O(|\mathcal{A}|^2\alpha)$, for a grand total of $O(|\mathcal{A}|^2\alpha N)$ for the entire regression. Clearly, $|\mathcal{A}|$ (i.e., the size of the output alphabet) is the dominating factor.

The worst-case storage requirements of digital locally monotonic regression are $O(|\mathcal{A}|\alpha N)$, but actual storage requirements are much more modest, due to path merging.

The availability of VLSI Viterbi decoding chips, as well as several dedicated multiprocessor architectures for Viterbi-type decoding, makes fast digital locally monotonic regression a realistic alternative to standard nonlinear filtering, at least for moderate values of $|\mathcal{A}|, \alpha$. In the binary case, current Viterbi technology [15], [16], [17], [18], [19] can handle 2^{12} states. Hardware capability is continuously improving, and at a rather healthy pace. Viterbi-type filtering techniques, like the one described here, will certainly benefit from these developments.

IV. Simulation Example

Let us now present a complete simulation experiment. Figure 2 depicts a typical input sequence. This particular input has been generated by adding i.i.d. noise on some artificial "true" noise-free test data, depicted in Figure 1. The noise has been generated according to a uniform distribution, and most of the data points are contaminated. It should be stressed that this is a "distribution-free" experiment, in that we do not use our prior knowledge of the noise model to match the regression to the noise characteristics, which is certainly a possibility (cf. [1]: by proper choice of $d_n(\cdot,\cdot)$, locally monotonic regression can be tailored to provide Maximum Likelihood (ML) estimates). The noise-free test data of Figure 1 is also overlaid on subsequent plots. This is meant to help the reader judge filtering "quality". Visual perception is arguably the ultimate "gold standard", and the reader is encouraged to attempt to trace the underlying signal visually.

For this example, we take $d_n(y(n), x(n)) = |y(n)-x(n)|$, $\forall n \in \{0, 1, \cdots, N-1\}$, $\mathcal{A} = \{0, \cdots, 99\}$, and $N = 512$. The resulting optimal approximation for $\alpha = 5, 10, 15, 20, 25$ is depicted in Figures 3, 4, 5, 6, and 7, respectively. The results look very promising. The overall run time is approximately equal to 2 minutes for $\alpha = 15$, $N = 512$, $|\mathcal{A}| = 100$, on a SUN SPARC 10, using simple C-code developed by the author, which certainly leaves much to be desired in terms of efficiency. Much better benchmarks may be expected for smaller alphabets and/or by implementing the algorithm in dedicated Viterbi hardware; e.g., for $|\mathcal{A}| = 32$, and everything else as above, the overall run time is approximately 12 seconds, for a throughput of 42 32-ary symbols per second.

V. Conclusions and Further Research

Motivated in part by the work of Restrepo and Bovik [1], our own earlier work in [8], and the fact that, in practice, one usually deals with digital (finite-alphabet) data, we have posed the problem of digital locally monotonic regression, in which the output symbols are drawn from a finite alphabet, as a natural optimal counterpart of median filtering of digital signals. Capitalizing on a connection between optimal nonlinear filtering under local syntactic constraints and Viterbi decoding algorithms, which has first been made in [8], we have provided a fast $O(|\mathcal{A}|^2 \alpha N)$ algorithm that computes any such regression, where $|\mathcal{A}|$ is the size of the digital output alphabet, α stands for lomo-degree, and N is sample size. This is *linear* (as opposed to *exponential* in the work of Restrepo and Bovik) in N, and it renders the technique applicable in practice.

The connection between optimal nonlinear filtering under local syntactic constraints and Viterbi decoding algorithms seems to be strong and pervasive; it appears to provide a unifying framework for the efficient computation of a rich class of nonlinear filtering techniques, some of which were oftentimes deemed impractical, due to their complexity. This key element certainly deserves further investigation, and several threads are currently being pursued.

VI. Acknowledgments

This research has been supported in part by core funds from the NSF ERC program, made available through the Communications and Signal Processing Group of the Institute for Systems Research of the University of Maryland; and industry, through Martin-Marietta Chair in Systems Engineering funds.

References

[1] A. Restrepo and A. C. Bovik, "Locally Monotonic Regression", *IEEE Trans. Signal Processing*, vol. 41, no. 9, pp. 2796–2810, Sep. 1993.

[2] N.C. Gallagher Jr. and G.W. Wise, "A theoretical analysis of the properties of median filters", *IEEE Trans. ASSP*, vol. ASSP-29, pp. 1136–1141, Dec. 1981.

[3] B. I. Justusson, "Median filtering: statistical properties", in *Two-Dimensional Digital Signal Processing II: Transforms and Median Filters*, T. S. Huang, Ed., pp. 161–196. Springer-Verlag, Berlin, 1981.

[4] T. A. Nodes and N. C. Gallagher, "Median filters: some modifications and their properties", *IEEE Trans. ASSP*, vol. ASSP-30, pp. 739–746, 1982.

[5] N.C. Gallagher Jr., "Median filters: a tutorial", in *Proc. IEEE Int. Symp. Circ., Syst., ISCAS-88*, 1988, pp. 1737–1744.

[6] A. C. Bovik, T. S. Huang, and D. C. Munson, "A generalization of median filtering using linear combinations of order statistics", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 31, pp. 1342–1349, 1983.

[7] A.C. Bovik, T.S. Huang, and D.C. Munson Jr., "The effect of median filtering on edge estimation and detection", *IEEE Trans. PAMI*, vol. PAMI-9, pp. 181–194, Mar. 1987.

[8] N.D. Sidiropoulos, "The Viterbi Optimal Runlength-Constrained Approximation Nonlinear Filter", Accepted for publication, IEEE Trans. Signal Processing. Also available as Institute for Systems Research ISR TR 95-45; and University of Maryland at College Park, OTL Disclosure IS95-27.

[9] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, N.J., 1957.

[10] R. Bellman and S. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton, N.J., 1962.

[11] S. Dreyfus and A. Law, *The Art and Theory of Dynamic Programming*, Academic, New York, NY, 1977.

[12] A.J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", *IEEE Trans. Information Theory*, vol. IT-13, pp. 260–269, Apr. 1967.

[13] J.K. Omura, "On the Viterbi decoding algorithm", *IEEE Trans. Information Theory*, vol. IT-15, pp. 177–179, Jan. 1969.

[14] B. Sklar, *Digital Communications*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[15] G. Feygin, P.G. Gulak, and P. Chow, "A multiprocessor architecture for Viterbi decoders with linear speedup", *IEEE Trans. Signal Processing*, vol. 41, no. 9, pp. 2907–2917, Sep. 1993.

[16] P.G. Gulak and E. Shwedyk, "VLSI structures for Viterbi receivers: Part I - general theory and applications", *IEEE J. Selected Areas in Communications*, vol. 4, pp. 142–154, Jan. 1986.

[17] S. Kubota, S. Kato, and T. Ishitani, "Novel Viterbi decoder VLSI implementation and its performance", *IEEE Trans. Communications*, vol. 41, no. 8, pp. 1170–1178, Aug. 1993.

[18] K.K. Parhi, "High-speed VLSI architectures for Huffman and Viterbi decoders", *IEEE Trans. Circuits and Systems II*, vol. 39, no. 6, pp. 385–391, June 1992.

[19] T.K. Truong, M.T. Shih, I.S. Reed, and E.H. Satorius, "A VLSI design for a trace-back Viterbi decoder", *IEEE Trans. Communications*, vol. 40, no. 3, pp. 616–624, Mar. 1992.

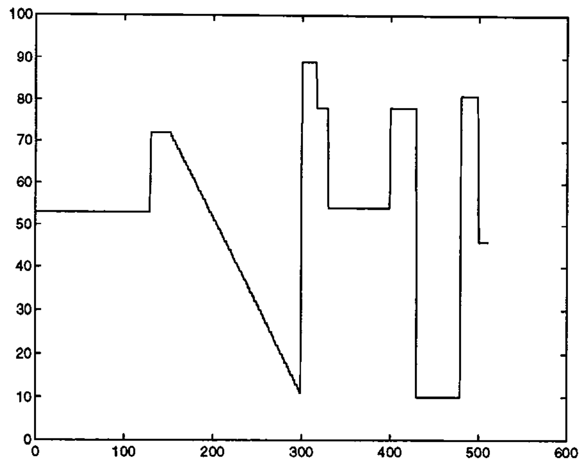

Fig. 1. The "true" noise-free test data

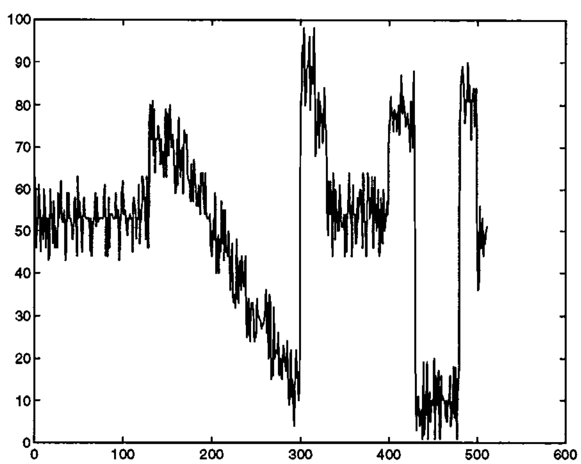

Fig. 2. Input sequence, $\{y(n)\}_{n=0}^{511}$

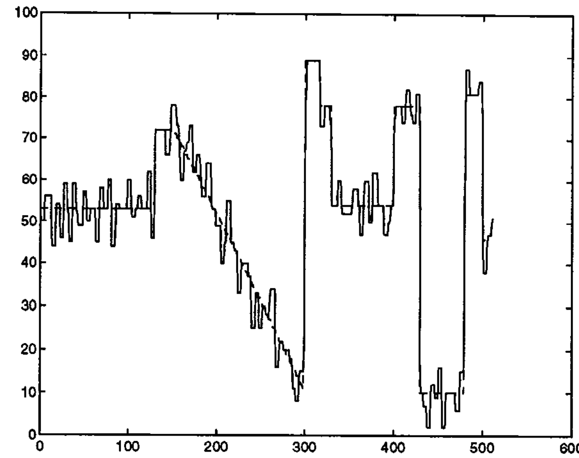

Fig. 3. Output of digital locally monotonic regression of degree $\alpha = 5$.

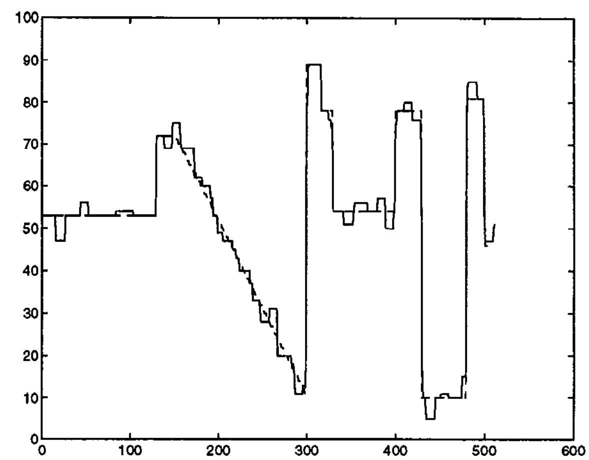

Fig. 4. Output of digital locally monotonic regression of degree $\alpha = 10$.

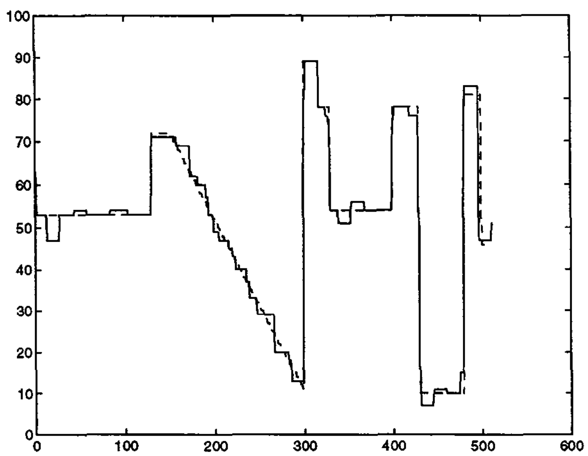

Fig. 5. Output of digital locally monotonic regression of degree $\alpha = 15$.

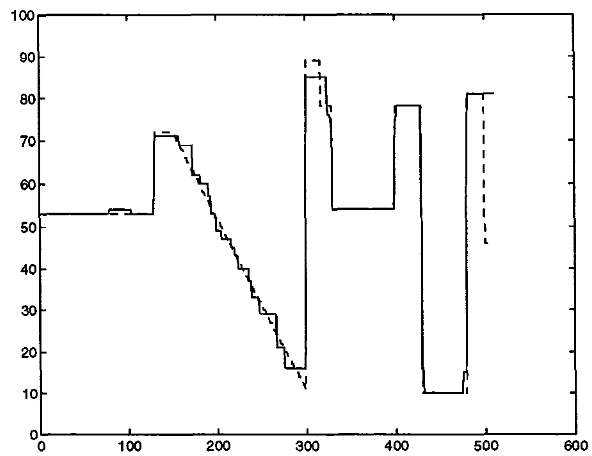

Fig. 7. Output of digital locally monotonic regression of degree $\alpha = 25$.

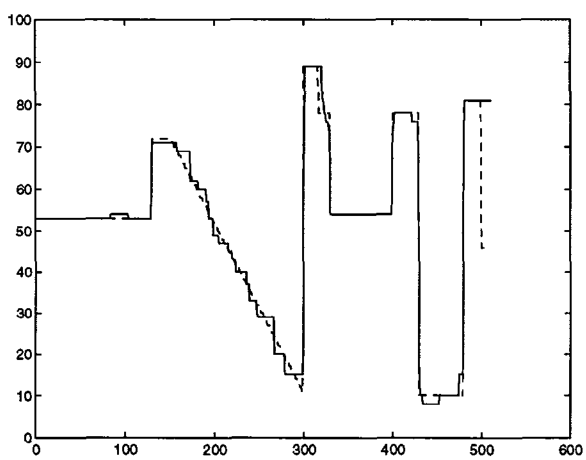

Fig. 6. Output of digital locally monotonic regression of degree $\alpha = 20$.

	$\alpha = 5$	$\alpha = 10$	$\alpha = 15$	$\alpha = 20$	$\alpha = 25$	$\alpha = 30$
$\|\mathcal{A}\| = 2$	26	56	86	116	146	176
$\|\mathcal{A}\| = 16$	1328	2688	4048	5408	6768	8128
$\|\mathcal{A}\| = 32$	5216	10496	15776	21056	26336	31616
$\|\mathcal{A}\| = 64$	20672	41472	62272	83072	103872	124672
$\|\mathcal{A}\| = 128$	82304	164864	247424	329984	412544	495104
$\|\mathcal{A}\| = 256$	328448	657408	986368	1315328	1644288	1973248

TABLE I

NUMBER OF DISTANCE CALCULATIONS AND ADDITIONS PER SYMBOL (I.E., PER TRELLIS STAGE). THE NUMBER OF COMPARISONS IS ALWAYS LESS THAN THIS NUMBER, AND THE COMPUTATIONAL COMPLEXITY PER TRELLIS STAGE IS ALWAYS LESS THAN TWICE THIS NUMBER.