Approximations of One-Dimensional Digital Signals Under the l^∞ Norm

Marco Dalai, *Student Member, IEEE*, and Riccardo Leonardi, *Member, IEEE*

***Abstract*—Approximation of digital signals by means of continuous-time functions is often required in many tasks of digital to analog conversion, signal processing, and coding. In many cases the approximation is performed based on an l^2 optimality criterion; in this paper we study approximations of one-dimensional signals under the l^∞ norm. We first introduce approximations in linear spaces, for which linear programming methods are known. For the particular case of linear approximations (i.e., first-order polynomials), we propose a geometric solution that is shown to be computationally more efficient than the linear programming approach. Then, we study the problem of piecewise approximations, i.e., dividing the domain into intervals and approximating the signal in linear spaces within every segment independently, so as to reach an optimal noncontinuous approximation. Given an error bound δ, we establish a strategy to determine the minimum number k of segments for which the approximation is guaranteed to produce an error within δ. We then show how to find the optimal partition that gives the piecewise l^∞ optimal solution with k segments. The computational complexity of the algorithms is studied, showing that in many practical situations, the number of operations is $O(n)$, with n being the number of samples.**

***Index Terms*—Linear programming, l-infinity approximations, minimum path, Viterbi algorithm.**

I. Introduction

APPROXIMATION of discrete signals by means of continuous time functions has been studied extensively in the literature (see, for example, [1]–[3] and references therein for an overview). Most of the attention has been dedicated to approximations under the l^2 norm, which means that the goodness of the approximation is established by evaluating the mean square value of the error. In this paper, we study approximations under the l^∞ norm.

Given a discrete set of points $D = \{x_i\}, x_i \in \mathbb{R}$, we consider a discrete signal s as a function $s : D \to \mathbb{R}$ that associates a real valued $s(x_i)$ to each value x_i in D. We indicate with l^∞ the set of functions f bounded over D and with $||\cdot||_\infty$ the norm defined, for $f \in l^\infty$, by

$$||f||_\infty = \sup_{x \in D} |f(x)| .$$

As usual, the distance between two functions f_1 and f_2 is then defined as the norm of the difference function, i.e., $d_\infty(f_1, f_2) = ||f_1 - f_2||_\infty$. The problem of approximating a signal s under the l^∞ norm consists of finding a function g from a given set of functions G such that the approximation error, that is, the distance $d_\infty(g, s)$, satisfies some given constraint. In some cases we will be interested in finding g which approximates s with an error smaller than a given threshold δ, while in other cases we will want g to be the function of G that minimizes this error.

In the following sections, we will study some of these particular problems, analyzing in detail some rather interesting special cases. This paper is organized as follows. In Section II, we briefly present the important case of approximations in linear spaces; we summarize the current approach for finding the optimal approximation, which reduces to solving a linear program, and we show how to use this technique for the solution of a more general approximation problem. For this section, we refer the reader to [2] for a detailed analysis of the approximation problem and to [4] for a general study of linear and nonlinear programming theory, even if it is not necessary for the understanding of this paper. In Section III, we consider the particular case of straight line approximations; we propose an efficient geometric algorithm for finding the optimal solution, showing the computational advantage of this method over the currently best performing linear programming technique. Then, in Section IV, we consider the problem of piecewise approximations in linear spaces. We show how to partition a given signal into a minimum number of segments so as to obtain a piecewise approximation within a given tolerance; we then show how to optimize the partition so as to minimize the error with the same number of segments. Finally, in Section V, we analyze the case of straight line piecewise approximations presenting a more efficient procedure based on the results of Section III. For a deeper analysis of computational geometry and optimization techniques used in Sections III–V, we refer the reader to [5]–[7, Sec. VI], but still this is not necessary for the understanding of the proposed methods.

II. Linear Spaces and Linear Programming

A very important special case of approximation problems is obtained when the domain D contains only a finite number n of points $x_i, i = 1, \ldots, n$, and the set G is a linear space generated from a finite set of basis functions. If $m \leq n$ is a fixed integer, we take a set $B = \{b_1, b_2, \ldots, b_m\}$ of m linearly independent functions b_j,[1] and we consider the set $G = \mathrm{span}(B)$; this means

[1]Here the term "linearly independent" will mean that any nontrivial linear combination of the b_j functions cannot be null over every point of D.

Manuscript received September 24, 2004; revised July 21, 2005. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Sheila S. Hemami.

The authors are with the Department of Electronics for Automation, University of Brescia, I-25123 Brescia, Italy (e-mail: marco.dalai@ing.unibs.it; riccardo.leonardi@ing.unibs.it).

Digital Object Identifier 10.1109/TSP.2006.875394

that for every $g \in G$, there exists a sequence of coefficients c_j such that

$$g = \sum_{j=1}^{m} c_j b_j. \tag{1}$$

This hypothesis implies that every possible approximation g to the signal s is uniquely identified by a sequence of real coefficients that are its representation in the B basis. Now, for uniformity with the literature, let us map every function g of G in the vector $\mathbf{g}$ of $\mathbb{R}^n$ whose ith component is the value $g(x_i)$, so as to work in a subspace over $\mathbb{R}^n$ instead of the space G.

If $\mathbf{A} \in \mathbb{R}^{n \times m}$ is the matrix with elements $a_{ij} = b_j(x_i)$, (1) is mapped to

$$\mathbf{g} = \mathbf{A}\mathbf{c}.$$

Thus, if we want to find the optimal l^∞ approximation of a signal s in F, we have to find the coefficient vector $\mathbf{c} \in \mathbb{R}^m$ that minimizes the value

$$\|\mathbf{s} - \mathbf{A}\mathbf{c}\|_\infty. \tag{2}$$

This problem has been studied extensively in the mathematical and mathematical programming literature (an exhaustive overview can be found in [2]; see [8] for a classic result) and the most recent approach consists in converting it to a linear program in $m+1$ dimensions. Accordingly, let $\mathbf{u} \in \mathbb{R}^n$ be the vector with all its components equal to one; then, for every fixed $\mathbf{c}$, the value in (2) is given by the smallest possible value of e, say, $e^*(\mathbf{c})$, that satisfies

$$-e\mathbf{u} \leq \mathbf{s} - \mathbf{A}\mathbf{c} \leq e\mathbf{u}$$

where inequalities between vectors are to be intended, here and in what follows, component by component. Subsequently, minimizing (2) is equivalent to minimize $e^*(\mathbf{c})$ as a function of $\mathbf{c}$.

If we set $\mathbf{d} = (\mathbf{c}, e)$ and we call $\mathbf{e}_{m+1}$ the $(m+1)$th vector of the canonical base of $\mathbb{R}^{m+1}$ (i.e., the vector whose $(m+1)$th component is equal to one and all other components are zero), we are minimizing the linear function

$$z = \mathbf{e}_{m+1}^T \mathbf{d}$$

under the conditions

$$\begin{bmatrix} \mathbf{A} & \mathbf{u} \\ -\mathbf{A} & \mathbf{u} \end{bmatrix} \mathbf{d} \geq \begin{bmatrix} \mathbf{s} \\ -\mathbf{s} \end{bmatrix}.$$

This formulation is exactly the enunciate of a linear programming problem in $m+1$ dimensions. Thus, for finding the solution of the approximation problem, it is possible to take advantage of the most advanced linear programming techniques that are available in the literature. In our case, however, it is particularly interesting to note that, if the parameter m can be considered fixed and much smaller than n, it is possible to solve the problem in $O(n)$ expected operations, as shown in [9] and [10] (see also [11, ch. 9]). In the following, we will always consider the parameter m to be constant, and we will thus assume that in linear spaces it is possible to compute the optimal l^∞ approximation in linear time (with respect to the number n of samples).

It is interesting to note that the idea of l^∞ approximation can be extended to a more general approach. Suppose, indeed, that we are still interested in controlling the approximation error at every point, as in l^∞ approximations, but assigning different weights (or, more precisely, different offsets) to different domain coordinates, i.e., approximate s with a maximum error that differs from point to point. Formally, this is expressed by stating that we want to find a function g such that

$$|s(x_i) - g(x_i)| \leq t(x_i), \quad i = 1, \ldots, n \tag{3}$$

where $t(x_i)$ is the allowed error in the point x_i. Interestingly, this problem can be treated in the same way, by using the additional variable e and minimizing e subject to the constraints

$$|s(x_i) - g(x_i)| \leq t(x_i) + e, \quad i = 1, \ldots, n. \tag{4}$$

In this case, clearly, we aim at finding a nonpositive value of e, thus verifying if the problem is feasible or not [i.e., a function g satisfying (3) exists]. If a positive value is obtained, we conclude that the problem is not feasible but having reached the knowledge of how far we are beyond the tolerance t. If, instead, a negative value of e is obtained, we know that the problem is feasible and we find the approximation that maximizes the margin from the threshold. In the latter case, however, it is important to note that the minimum value of e cannot be less than $-\min_i(t(x_i))$, as the values on the right-hand side of (4) must be nonnegative. Thus, by calling x_m the point in which t reaches the minimum, in some cases it is possible to fit s in x_m while still satisfying (3) for every other point x_i. In this case, in the linear program, we have a constraint (given by the point x_m, i.e., $|s(x_m) - g(x_m)| \leq t(x_m) + e$) that is orthogonal to the minimization vector and thus the solution is not unique. In this situation it could be convenient to project the problem into the hyperplane $s(x_m) - g(x_m) = 0$, so as to optimize the approximation over $x_{i \neq m}$ while imposing exact interpolation in x_m. We conclude by clarifying that a program for the classical l^∞ approximation can be used for this more general type of approximation. In fact, let d be a constant such that $d > \|t\|_\infty$. Thus, if we set $s^+(x) = s(x) - t(x) + d$ and $s^-(x) = s(x) + t(x) - d$, it is easy to see that (3) is equivalent to

$$\begin{cases} |s^+(x_i) - g(x_i)| \leq d \\ |s^-(x_i) - g(x_i)| \leq d \end{cases} \quad i = 1, \ldots, n.$$

Thus, the approximation of s with a variable tolerance t can be obtained by approximating s^+ and s^- jointly with the usual l^∞ norm. This idea has been of practical utility, in the field of

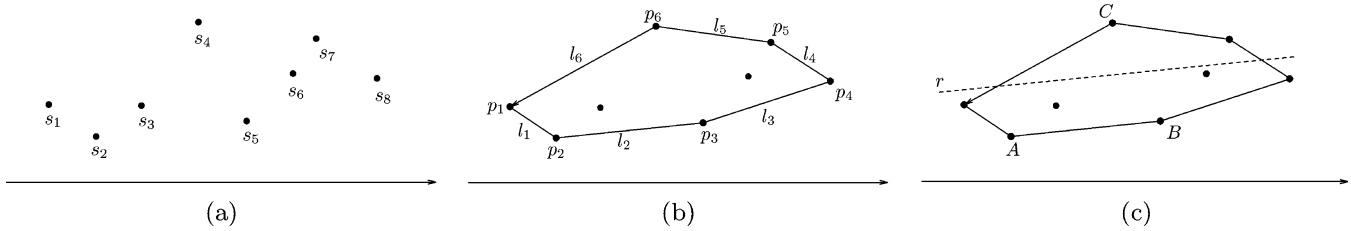


Fig. 1. Steps of the geometric method for single link optimal solution. (a) Signal samples. (b) Convex hull found. (c) Solution found.

image coding, in [12], where a separable approach has been used for the problem of finding bidimensional l^∞ suboptimal bilinear approximations. We refer the reader to [12] for more details.

III. Linear Approximations

In this section, we study the particular case of linear approximations, i.e., by means of first-order polynomials. In this case the general function g of G is expressed as $g(x) = ax + b$, where a and b are real numbers. It is clear that in this case we can take $B = \{1, x\}$; thus the space G has dimension 2 and the problem of finding the best approximation of a given signal s is equivalent to solving a three-dimensional linear program. In particular, we have to minimize the linear function

$$z = [0, 0, 1] \begin{bmatrix} a \\ b \\ e \end{bmatrix}$$

under the constraints

$$\begin{bmatrix} x_1 & 1 & 1 \\ x_2 & 1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & 1 & 1 \\ -x_1 & -1 & 1 \\ -x_2 & -1 & 1 \\ \vdots & \vdots & \vdots \\ -x_n & -1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ e \end{bmatrix} \geq \begin{bmatrix} s(x_1) \\ s(x_2) \\ \vdots \\ s(x_n) \\ -s(x_1) \\ -s(x_2) \\ \vdots \\ -s(x_n) \end{bmatrix}.$$

For what has been said at the end of the previous section, very efficient linear programming techniques are available for this problem, and the optimal approximation can be found in $O(n)$ expected number of operations. The expectation is due to the fact that such linear programming techniques are based on randomized methods, and thus the number of operation used for a fixed signal is a random variable. We propose here a geometry-based algorithm that can outperform the linear programming technique by exploiting the particular nature of the problem. The advantages of this algorithm will be detailed; we only remark here that it is deterministic and uses $O(n)$ operations in the worst case, as we will prove in what follows.

A. The Proposed Geometric Algorithm

Let $D = \{x_i\}_{i=1,\ldots,n}$ be the domain of n points of $\mathbb{R}$, s be the signal, and S be the set of n points s_i of the signal samples in the plane, i.e., $s_i = (x_i, s(x_i))$. Let Q be the convex hull of S, that is, the smallest convex polygon that contains every point of S. Let us define some notations for clarity. Let k be the number of sides of Q; we indicate with p_i, $i = 1, \ldots, k$, the vertices of Q in counterclockwise order, with p_1 the leftmost one. For convenience, we add a new point $p_{k+1} = p_1$; then we indicate with l_i, $i = 1, \ldots, k$, the side $\overline{p_i p_{i+1}}$. Let m be the integer such that p_m is the rightmost vertex of Q; then we will call *lower hull* the polygonal line formed by the sides l_i, $i = 1, \ldots, m-1$, and *upper hull* the polygonal line formed by the sides l_i, $i = m, \ldots, k$. We will consider that the vertices p_1 and p_m belong to both the upper and lower hull. Finally, given a side l and a point p, we will say that p is x-internal to l if the vertical line through p cuts the side l; on the contrary, we will say that p is x-external on the left or on the right, the meaning being obvious.

Now, suppose for a moment, for simplicity of the presentation, that Q has no pair of parallel sides. Then, to each side l of Q it is possible to find a vertex $v(l)$ of Q that is the most distant one from l in the orthogonal direction; we call $v(l)$ the *opposite vertex* to the side l.

Proposition 1: Under the above hypothesis, there exists one and only one side l of Q such that $v(l)$ is x-internal to l. The optimal linear approximation of the signal s over the domain D is then the line r parallel to l and equidistant from l and $v(l)$. (For a proof, see Appendix A). We call the extremities A and B, $A < B$, of the side l and the opposite vertex $C = v(l)$ pivot points of the set S, so as to identify the three points that determine the optimal linear approximation.

This proposition gives a very useful property of the geometry of the polygons. By using this proposition, we can construct a very efficient geometric algorithm to find the l^∞ optimal linear approximation of a signal s (see Fig. 1).

Algorithm 1

- Compute the convex hull of the set S.
- Scan the sides of the convex hull computing their opposite vertex until the pivot points A, B, and C are found.
- Compute the solution line r.

We now give a detailed explanation of the first two steps of Algorithm 1 (the third step is only a simple computation), for which we propose efficient subalgorithms showing that the number of operations is $O(n)$.

1) Computing the Convex Hull: Finding the convex hull of a set of points in the plane is one of the most studied problems of computational geometry, and several algorithms are available

for this task (see [5] and [13]). One important thing to be considered here is that the points are sorted with respect to the x coordinate. Under this hypothesis, it is possible to find the convex hull of the set S in $O(n)$ operation using Graham's algorithm [14]. Here we recall only that the main idea is to construct the convex hull by moving from left to right; at every step the polygon is updated by adding a new point and removing the sides of the polygon that are visible by the entering point. The only basic operation that is required for this algorithm is the evaluation of the order of three generic points q_1, q_2, and q_3 in the plane,[2] and it is easy to see that this evaluation is nearly equivalent to the evaluation of the vector product $\overrightarrow{q_1q_2} \times \overrightarrow{q_2q_3}$. (See Fig. 2 for a graphical explanation. See also [7, Sec. 33.3] for a detailed description of these operations.)

2) Finding the Vertices A, B, and C: Once we have constructed the convex hull Q of the set S, we have to search the side l such that its opposite vertex $v(l)$ is x-internal to l. We now state some simple lemmas that suggest an efficient way to find the searched l and $v(l)$. These lemmas are proved in the Appendix, where they are also used for the proof of Proposition 1.

Lemma 1: Every side of the lower hull has its opposite vertex in the upper hull and vice versa.[3]

Lemma 2: If we move from one side of the polygon to its consecutive in counterclockwise (CCW) direction, the respective opposite vertex, if it changes, moves in CCW direction too.

Lemma 3: A vertex p_j, $1 < j \leq k$, is the opposite vertex of a side l_i, i.e., $p_j = v(l_i)$, if it is more distant from l_i than the vertices p_{j-1} and p_{j+1}.

These considerations lead to a good algorithm for finding the opposite vertex of each side of the lower hull, and thus also the searched l and $v(l)$. As a general notation, we call j_i the integer such that $p_{j_i} = v(l_i)$.

Algorithm 2

- Find the opposite vertex of l_1: starting from p_m scan in CCW direction the vertices of the upper hull, computing their distances from l_1 until we find a vertex p_{j_1+1} which is less distant from l_1 than p_{j_1}. Thus $v(l_1) = p_{j_1}$.
- Continue by considering the sides of the lower hull to find their opposite vertices. For each side l_i we have to control the vertices of the upper hull from $v(l_{i-1})$ (in CCW direction) until we find a vertex p_{j_i+1} that is less distant from l_i than p_{j_i}. Then $v(l_i) = p_{j_i}$.
- Do the same, symmetrically, for the upper hull sides.

Proposition 2: Algorithm 2 requires at most $3k$ vector product computations[4] of the type $\overrightarrow{p_ip_{i+1}} \times \overrightarrow{p_{i+1}p_j}$ for finding the opposite vertices of all convex-hull sides.

Proof: Consider for a moment only the number of distance computations required for finding the opposite vertices of the lower hull sides. Consider the generic side l_i and suppose we have found the opposite vertex $p_{j_{i-1}}$ of the previous side l_{i-1}, i.e., $p_{j_{i-1}} = v(l_{i-1})$. It is easy to see that for finding the opposite vertex p_{j_i} of l_i, one must compute $2 + (j_i - j_{i-1})$ distances. For example, suppose $v(l_2) = p_8$ and $v(l_3) = p_{10}$; if $p_8 = v(l_2)$, in order to find $v(l_3)$ one has to compute the distances of p_8, p_9, p_{10}, and p_{11} from l_3, and thus $4 = 2+(10-8)$ distances. For the first side l_1, the same argument holds setting $j_0 = m$ as in this case, we start to check the vertices starting from p_m. This means that the total number of computed distances is $\sum_{i=1}^{m-1}(2 + j_i - j_{i-1}) = 2(m-1) + (j_{m-1} - m)$. But clearly $j_{m-1} \leq k+1$ and thus the opposite vertices of the sides of the lower hull are found by computing at most $k+m-1$ distances. Considering the symmetry of the problem, we can say that the opposite vertices of the upper hull sides can be found by computing at most $2k - m+1$ distances, for a total of at most $3k$ distance evaluations. It is clear, however, that the algorithm will stop, for the problem of interest, when the side l with opposite x-internal vertex has been found. Finally, we now show that in fact one does not need to compute $3k$ distances but only $3k$ vector products of the type $\overrightarrow{p_ip_{i+1}} \times \overrightarrow{p_{i+1}p_j}$, which represent a smaller computational cost. In fact, when searching the opposite vertex of the generic side l_i, we do not need to really know the distances of the generic point p_j from l_i, but only compare the values for different j. Considering that the distances of p_j from l_i (for varying j but fixed i) are proportional to the areas of the triangles of vertices p_i, p_{i+1}, and p_j, we can compare the value of these areas instead of the distances. Since twice the area of the triangle of vertices p_i, p_{i+1}, and p_j equals the vector product $\overrightarrow{p_ip_{i+1}} \times \overrightarrow{p_{i+1}p_j}$, this implies that the algorithm requires only $3k$ such vector products. ■

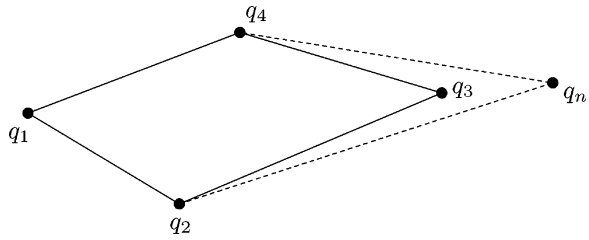


Fig. 2. Example of construction of the convex hull with Graham's method. The point q_n has to be inserted after q_2 since $\overrightarrow{q_1q_2} \times \overrightarrow{q_2q_n} > 0$ while $\overrightarrow{q_2q_3} \times \overrightarrow{q_3q_n} < 0$. Symmetrically, q_n has to be inserted before q_4 since $\overrightarrow{q_4q_3} \times \overrightarrow{q_3q_n} > 0$ while $\overrightarrow{q_1q_4} \times \overrightarrow{q_4q_n} < 0$.

Going back to Algorithm 1, it can be stated that this algorithm requires $O(n)$ operations and that the only required basic function is the evaluation of vector products. Whereas from a theoretical point of view Algorithm 2 is quite useful, it can be further improved by using the following property of a convex hull Q.

Lemma 4: Given two consecutive sides l_i and l_{i+1}, their common vertex p_{i+1} is the opposite vertex of every side between $v(l_i)$ and $v(l_{i+1})$ (in the path not containing l_i and l_{i+1}, obviously). With such a consideration, Algorithm 2 can be modified as follows.

[2]The order of three points q_1 q_2 and q_3 is defined to take value: 0 if the three points are aligned, 1 if the oriented polygonal $q_1 \rightarrow q_2 \rightarrow q_3$ turns counterclockwise and -1 if it turns clockwise.

[3]Remember that p_1 and p_m belong to both the upper and lower hull.

[4]Recall that k is the number of sides of the convex hull.

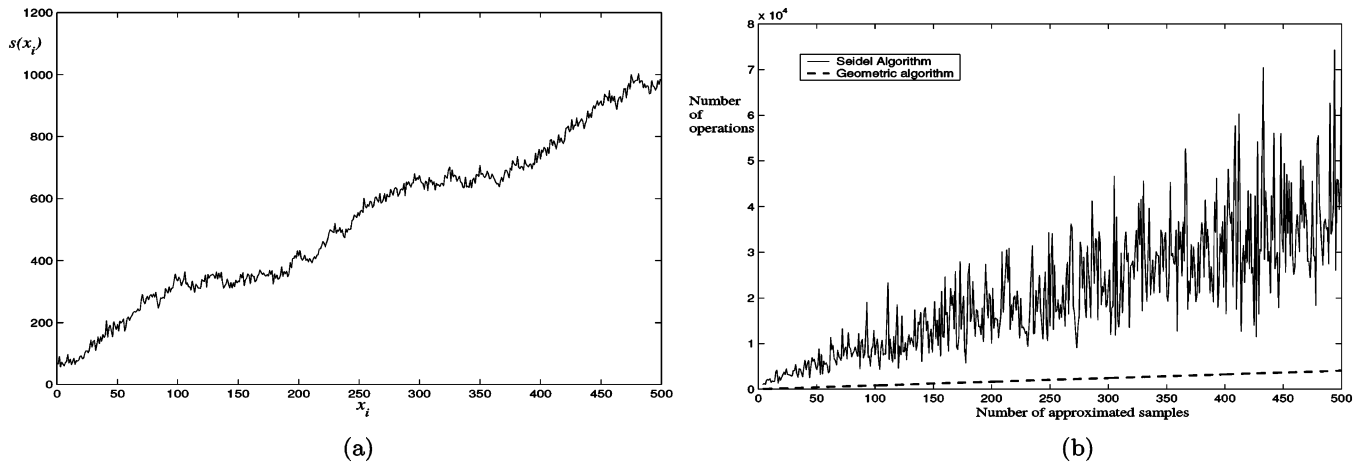


Fig. 3. Comparing the number of operations used by the geometric method and Seidel's randomized algorithm for linear programming in small dimensions. (a) Signal used for approximations. The near-linear behavior has been obtained by summing small sinusoidal functions and white Gaussian noise to a straight line. (b) Number of operations used by the two algorithms.

Algorithm 3

- Find $v(l_1)$, opposite vertex of l_1, as suggested in Algorithm 2.
- For $i \geq 1$, once $v(l_i)$ is found, check
  a) if $v(l_i)$ is x-external to l_i on the right go on searching $v(l_{i+1})$;
  b) else if $v(l_i)$ is x-internal to l_i terminate the search;
  c) else if $v(l_i)$ is x-external to l_i on the left, search the side of the upper hull between $v(l_{i-1})$ and $v(l_i)$ such that the common vertex of l_{i-1} and l_i, i.e., p_i, is x-internal to it.

It is important to note that this algorithm is strongly based on the proof of Proposition 1; this ensures that one of the items b) or c) is reached before finishing scanning the sides of the lower hull and thus the algorithm always finds the solution. With this algorithm the number of computed vector products is reduced by about a factor of two in the mean case with respect to the performance of Algorithm 2.

B. Performance Comparison

Compared to the linear programming solution, the geometric algorithm has many advantages. The first one is that it is very easy to implement and generates a very compact code. As it has been shown, all the computations in the construction of the convex hull and the scanning of its side-vertices pair can be reduced to a vector product operation; thus, the implementation requires a few loops calling a simple function for the computation of a vector product. Furthermore, as the vectors are always coplanar, this operation is only a sum of products of the type $\mathrm{e} = \mathrm{ab} + \mathrm{cd}$, which can be executed very efficiently on many digital signal processors. Moreover, the memory usage is very limited; the only memory space needed (apart from the input sequence) is a vector containing the indexes of the points that are vertices of the convex hull Q, which represent at most n integers. Furthermore, it is important to note that, if we are working with discrete signals, almost all the computations can be performed using only fixed-point arithmetic. The only need for floating-point operations is indeed due to the construction of the optimal line from the pivot points and the evaluation of the approximation error, which represent a fixed number of operations. This is a concern in case a floating-point unit is not available.

Finally, an important consideration is about the computation time. First of all, from a theoretical point of view, our algorithm has a computation time that is $O(n)$ in the worst case, while the linear programming techniques can only provide a solution in $O(n)$ on average. For practical considerations, then, we have compared our algorithm with an ad hoc implementation of the Seidel randomized algorithm for linear programming in small dimensions [9], which is known to be very fast for this kind of problems. For this purpose we have counted the number of operations used by the two algorithms, so as to remove any dependency on the machine architecture, type of data (we recall that our algorithm can work without using floating point arithmetics) and, most important, memory usage.[5] We have taken a signal with near-linear behavior, shown in Fig. 3(a), and we have computed the linear approximation of the first n points, with n varying from 4 to 500. The number of operations used by the two methods, as a function of n, is plotted in Fig. 3(b). As it can be seen, the Seidel algorithm presents an irregular behavior, due to its randomized nature, having linear complexity in the mean. The geometric method, instead, gives a regular increase of the number of operations, which leads to a speedup by a factor ranging from 3 to more than 15 with respect to the Seidel algorithm, with a mean gain of about 8.

IV. Piecewise Approximations With Error Bound

Often signal approximation in linear spaces is not a practical tool for signal processing and coding due to the fact that the

[5]It is also relevant to notice that, for a fast implementation of the Seidel algorithm, one should not make use of dynamic memory allocation; this implies the necessity of allocating more than $30n$ floating-point variables, against the n integers of the geometric algorithm. If instead one wants to reduce memory usage (in any case much more than n integers), memory should be allocated dynamically, thus leading to a significant reduction of the computational efficiency.

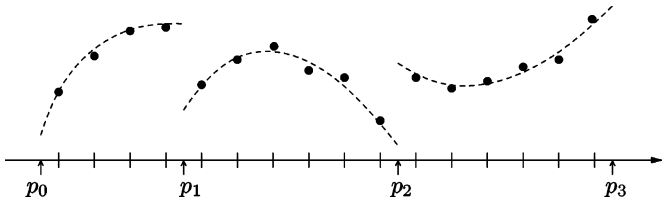


Fig. 4. Example of piecewise approximation of a 16-sample signal with associated partition points. Here $\nu(g) = 3, p_0 = 0.5, p_1 = 4.5, p_2 = 10.5$, and $p_3 = 16.5$.

dimension of the approximation space must increase with the number of samples if we want to keep small values of the error. Thus, it is necessary to divide the domain in smaller subdomains (intervals) such that the signal can be approximated with small error in a small dimensional space within every domain. With this idea, we define a more general function space: given a set of functions B (and the induced G) we call G_T the set of functions g over D for which there exists a partition of D in subdomains Δ_k such that the restriction of g to every Δ_k is in G.

In this section and in the next, we study the problem of optimizing the partition of the domain into connected subdomains while approximating the signal within an error threshold. Consider that, given a partition of the domain, the problem of approximating the signal within each subdomain is only an application of what has been described in Section II. So, in the following, the emphasis will be addressing mainly the partition of the domains. For the sake of clarity, we suppose that the set S is characterized by $x_i = i, i = 1, \ldots, n$, even though the presented results hold in the case of nonuniform samples. Given any piecewise approximation g of the signal, we characterize it with an error $e(g)$, a number of connected subdomains $\nu(g)$, and a *partition set* $P(g) = \{p_i(g)\}_{i=1,\ldots,\nu(g)-1}$ of values such that $p_i = m + 1/2$ if m is the last point of the ith interval and $m+1$ is the first point of the $(i+1)$st interval. Moreover, we set $p_0(g) = 1/2$ and $p_{\nu(g)}(g) = n + 1/2$, and it is implicitly considered that the partition points p_i can only take values of the type $m+1/2$ with $m \in \mathbb{N}$. See Fig. 4 for an example of piecewise approximation with the associated partition points. All considered intervals[6] are measured on a discrete half integer value. Thus we will identify the "approximation on the interval [3/2,7/2]" as "the one of locations 2 and 3"; similarly, by stating "the partition point p_i is in]3/2,7/2]" is equivalent to saying "$p_i \in \{5/2, 7/2\}$."

We now study the problem of optimally partitioning the domain by introducing the idea of minimal and optimal approximations for a given error threshold δ.

A. Minimal Solution

The problem to be solved is the following: given the set S of n samples of signal s, the set B of the basis functions, and an error bound δ, we want to find an approximation $g \in G_T$ of s with error $e(g) \leq \delta$ such that the number $\nu(g)$ of intervals is the smallest possible.

[6]We use bracket notation for intervals. So, $[a, b]$ is the interval containing both a and b, while $]a, b[$ contains none, $[a, b[$ contains a but not b, and $]a, b]$ contains b but not a.

In general there are more solutions to this problem, and we aim at finding at least one of them. Interestingly enough, it is possible to find two solutions (not necessarily distinct) with a very simple algorithm by scanning the signal in a progressive fashion.

For finding these solutions, we first need an algorithm that finds, given any point $s_k = (x_k, s(x_k))$, the "longest" possible approximation of s starting from it in one direction, e.g., the maximum value l such that the points $s_i = (x_i, s(x_i)), i = k, \ldots, k + l$ can be approximated in B with error smaller than the threshold δ. This leads to an approximation of the l points which is consistent with the error constraint δ. The algorithm is the following.

Algorithm 4

- Compute the optimal approximations over the intervals $[k, k+1], [k, k+2], \ldots, [k, k+2^j], \ldots$ until an error larger than δ is obtained (which happens for $j = \lceil \log_2 l \rceil$); call $a = \lceil \log_2 l \rceil$.
- Find l with a binary search on the interval $[2^{a-1}, 2^a[$.

Proposition 3: Algorithm 4 requires an expected number of $O(l \log l)$ operations.

Proof: Consider the first step of the algorithm; the optimal approximation over the generic interval $[k, k + 2^j]$ can be found in $O(2^j)$ expected operations using Seidel randomized algorithm, as explained in Section II. Thus, a is found in $\sum_{j=0}^{a} O(2^j) = O(2^{a+1})$ operations. Then, the second step of the algorithm computes at most a approximations of length less than or equal to $2l$. So, the expected number of operations for the second step is $O(a \cdot l) = O(l \log l)$, which is the dominating term. ■

We now apply the proposed algorithm for the construction of two approximations that we will prove to be minimal. We indicate these approximations with $\overrightarrow{g}$ and $\overleftarrow{g}$ so as to emphasize the fact that they are obtained by scanning the signal, respectively, from left to right and vice versa. Here we give the algorithm for finding $\overrightarrow{g}$.

Algorithm 5

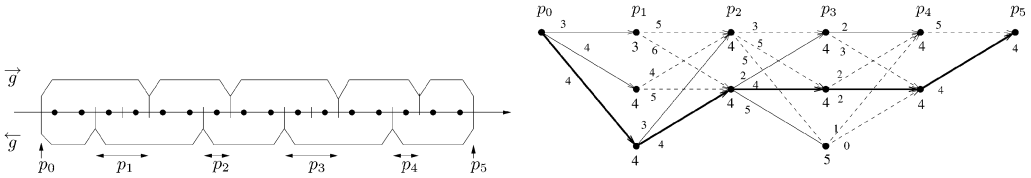
- Start by scanning the signal from the first point s_0. Using Algorithm 4, find the first longest possible approximation segment, and thus the partition point $p_1(\overrightarrow{g})$. Set i to 1.
- Given $p_i(\overrightarrow{g})$, compute the longest possible approximation segment (using Algorithm 4) starting from $p_i(\overrightarrow{g})$, and thus find $p_{i+1}(\overrightarrow{g})$. Repeat until the end of the signal is reached.

Proposition 4: Algorithm 5 requires an expected number of $O(n \log n)$ operations.

Proof: Note that, setting $l_i = p_i(\overrightarrow{g}) - p_{i-1}(\overrightarrow{g})$, from Proposition 3, we need $O(l_i \log l_i)$ operation for finding $p_i(\overrightarrow{g})$. Thus, we need $\sum_i O(l_i \log l_i)$ operations; considering that $\sum_i l_i = n$, we have $\sum_i O(l_i \log l_i) < \sum_i O(l_i \log n) = O(n \log n)$. ■

Example of Domain Partition

We show an example of the algorithm for finding the optimal partition set. A signal, which we suppose to have 16 samples, is first approximated as explained in paragraph IV-A, by scanning it from left to right and right to left, obtaining the two approximations $\overrightarrow{g}$ and $\overleftarrow{g}$. Suppose that these two partitions result as shown in fig. 5(a), so that $P(\overrightarrow{g}) = \{0.5, 4.5, 7.5, 11.5, 14.5, 16.5\}$ and $P(\overleftarrow{g}) = \{0.5, 2.5, 6.5, 9.5, 13.5, 16.5\}$. Thus, we have some restrictions on the possible positions of the optimal p_i, as shown in the figure. Now, we construct the trellis associated with these possible choices. So, we have three states for p_1 and p_3 (i.e. $w_1 = w_3 = 3$) and two states for p_2 and p_4 (i.e. $w_2 = w_4 = 2$). To find the optimal path in the trellis we flag every link with a weight given by the approximation error on the interval relative to that link (small numbers in Fig. 5(b)); then, moving from left to right, we flag the nodes with the *accumulated state metric* values (larger numbers on Fig 5(b)), as explained in section IV-B. In Fig. 5(b) the dashed links are those that have been discarded because of non-optimality while the solid ones are the optimal choices for their entering nodes. The bold path is overall the optimal one, as it is the only path from p_0 to p_5 that can be constructed with solid links only.



(a) Example of partitions of a domain

(b) Trellis equivalent to the partition example.

Fig. 5. Finding the optimal partition set. (a) Example of partitions of a domain. (b) Trellis equivalent to the partition example.

Clearly, the same procedure of Algorithm 5 can be used analyzing the signal from right to left to obtain the approximation that we denote with $\overleftarrow{g}$. It is also clear that both $\overrightarrow{g}$ and $\overleftarrow{g}$ depend on δ; we now show that they are indeed minimal for that given value of δ.

Proposition 5: The two approximations $\overrightarrow{g}$ and $\overleftarrow{g}$ lead to the same number of segments $k = \nu(\overrightarrow{g}) = \nu(\overleftarrow{g})$, and every approximation h such that $e(h) \leq \delta$ satisfies $\nu(h) \geq k$.

Proof: Consider the construction of $\overrightarrow{g}$. The way $p_1(\overrightarrow{g})$ was obtained implicitly says that it is not possible to approximate the interval $[1, p_1(\overrightarrow{g})+1]$ with a single segment (without exceeding the value of δ); h cannot be an exception and thus $P(h)$ must have a partition point $p_1(h)$ in the interval $[1, p_1(\overrightarrow{g})]$. Similarly it is not possible to approximate with a single segment the interval $[p_1(\overrightarrow{g}), p_2(\overrightarrow{g}) + 1]$, so that $P(h)$ must have at least another point $p_2(h)$ in $]p_1(\overrightarrow{g}), p_2(\overrightarrow{g})]$ as $p_1(h) \leq p_1(\overrightarrow{g})$. By iterating the argument, this proves by induction that for $1 \leq i \leq k-2$ there must exist a point of $P(h)$ in the interval $]p_i(\overrightarrow{g}), p_{i+1}(\overrightarrow{g})]$ and thus $\nu(h) \geq k$. In particular, setting $h = \overleftarrow{g}$, we obtain that $\nu(\overleftarrow{g}) \geq \nu(\overrightarrow{g})$; but, by symmetry of the construction process, in the same way we could prove that $\nu(h) \geq \nu(\overleftarrow{g})$ and thus, setting now $h = \overrightarrow{g}$, $\nu(\overrightarrow{g}) \geq \nu(\overleftarrow{g})$. This means that $\nu(\overrightarrow{g}) = \nu(\overleftarrow{g})$ and that this number k of intervals is minimal. ■

B. Optimal Solution

In the preceding section, we have seen how to find a piecewise approximation that uses the minimum number of intervals in $O(n \log n)$ operations. More specifically, we have seen that it is possible to find two solutions $\overrightarrow{g}$ and $\overleftarrow{g}$, each being minimal. Now, given that the number of used intervals cannot be further lowered, we can ask for the minimal approximation that minimizes the approximation error. For this task, we now show that the $\overrightarrow{g}$ and $\overleftarrow{g}$ solutions provide two partition sets that are a sort of extremes of the possible partition sets of any minimal approximation. More precisely, we have the following.

Proposition 6: If h satisfies $e(h) \leq \delta$ and $\nu(h) = k$, then, for every $1 \leq i \leq k-1$, we have $p_i(\overleftarrow{g}) \leq p_i(h) \leq p_i(\overrightarrow{g})$.

Proof: If $e(h) \leq \delta$, we have already proved (in the proof of Proposition 5) that there exists a point of $P(h)$ in $]p_i(\overrightarrow{g}), p_{i+1}(\overrightarrow{g})]$ for $0 \leq i \leq k-2$. If $\nu(h) = k$, then in each interval there is exactly one point, which has to be $p_{i+1}(h)$. This holds for $h = \overleftarrow{g}$ so that $p_i(\overrightarrow{g}) < p_{i+1}(\overleftarrow{g}) \leq p_{i+1}(\overrightarrow{g})$. By symmetry, we can say that if $e(h) \leq \delta$ and $\nu(h) = k$, there is exactly one point $p_i(h)$ in $[p_i(\overleftarrow{g}), p_{i+1}(\overleftarrow{g})[$ for $1 \leq i \leq k-1$ and, for $h = \overrightarrow{g}$, we obtain $p_i(\overleftarrow{g}) \leq p_i(\overrightarrow{g}) < p_{i+1}(\overleftarrow{g})$. By combining these inequalities we reach the result that if h is a k-link solution, then $p_i(\overleftarrow{g}) \leq p_i(h) \leq p_i(\overrightarrow{g})$ for every $1 \leq i \leq k-1$. ■

From now on we will call $w_i = p_i(\overrightarrow{g}) - p_i(\overleftarrow{g}) + 1$ the number of possible values that p_i can take, using the notation $p_i^j = p_i(\overleftarrow{g}) + j - 1, j = 1, \ldots, w_i$. Furthermore, we follow the notation of the previous section and set $l_i = p_i(\overrightarrow{g}) - p_{i-1}(\overrightarrow{g})$; thus $w_i \leq l_i$ for every value of i (see Fig. 6).

The above consideration provides a very important property of the possible partitions of the domain to obtain a minimal solution because it reduces enormously the number of possible choices of the partition points and gives then the possibility of efficiently finding the optimal solution. We show, in fact, that it is possible to reduce the problem of finding the optimal approximation to the problem of finding a *minimum path* in a graph. (See [7, Sec. VI] for a presentation of general graph algorithms. Also note later that our graph problem is not a minimum-path

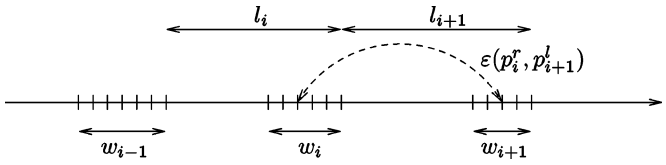


Fig. 6. Computational complexity of the evaluation the link weights $\varepsilon(p_i^r, p_{i+1}^l)$. For every fixed p_{i+1}^l, we can find the optimal p_i^r with a binary search, thus computing $\log_2(w_i)$ approximation. The number of points used in every approximation is at most $w_i + l_{i+1}$; clearly, for every i, we have $w_i \leq l_i$ and thus the number of expected operation in finding the optimal p_i^r for every fixed p_{i+1}^l is less than or equal to $(l_i + l_{i+1}) \log_2 l_i$. When all the p_{i+1}^l points are checked, the number of expected operation is then at most $(l_i + l_{i+1}) l_{i+1} \log_2 l_i$.

problem in the classic sense, as the metric is usually additive in such problems, but this has no practical implication.) Note that every piecewise approximation can be considered as a path between the first point and the last one, passing through some nodes that are represented by the partition points. Every interval is then represented as a link between two nodes, and we can associate to every link a cost given by the l^∞ approximation error over the corresponding interval. Considering that every minimal approximation has exactly one partition point in every interval $[p_i(\overleftarrow{g}), p_i(\overrightarrow{g})]$, we can represent the whole set of these approximations with a trellis graph of the type shown in Fig. 5(b), in which nodes on the same column represent possible positions of one single partition point. In this graph we have to search the path that goes from the first point to the last one minimizing the greatest value encountered on its links. It is not difficult to note that this problem can be solved with a variant of the well-known Viterbi algorithm [15]; the only difference is that, while the Viterbi algorithm is based on an additive metric, here we have to use the maximum metric. This means that the cost of a path in the trellis is not given by the sum of the costs of the single links, but by the maximum of their values. The idea is to find the optimum path proceeding from left to right. We label every node p_i^j with an *accumulated state metric* $e(p_i^j)$, which represents the cost of the optimal path from the point p_0 to it, and we establish its antecedent $a(p_i^j)$, which is the optimal choice of the point p_{i-1} for reaching p_i^j. In order to be more precise, calling $\varepsilon(p_i^j, p_{i+1}^l)$ the cost of the link connecting p_i^j to p_{i+1}^l, we give the detailed algorithm for finding the optimal path in the graph.

Algorithm 6

- Define, for $r = 1, \ldots, w_1$, the accumulated state metrics of the p_1^r points as $e(p_1^r) = \varepsilon(p_0, p_1^r)$ and the antecedent points as $a(p_1^r) = p_0$.
- Given the accumulated state metrics of the points p_i^r, $r = 1, \ldots, w_i$, define, for $l = 1, \ldots, w_{i+1}$, the value $e(p_{i+1}^l)$ as

$$e\left(p_{i+1}^l\right) = \min_r \left(\max\left(e\left(p_i^r\right), \varepsilon\left(p_i^r, p_{i+1}^l\right)\right)\right) \tag{5}$$

and $a(p_{i+1}^l) = p_i^{r_{\min}}$, where $r_{\min}$ is the value of r that gives the minimum in (5). Iterate this point until $p_{k+1} = n + 1/2$ is reached.

Consider the behavior of this algorithm. In the first step the accumulated metrics for the points p_1^r are set. Then, for every possible choice p_2^l of p_2, we select the point p_1^r such that the value $\max(e(p_1^r), \varepsilon(p_1^r, p_2^l))$ is the smallest possible. So, for every p_2^l, we keep only one point p_1^r and consequently one path that is the optimal choice for reaching p_2^l. Then we define the new accumulated metric $e(p_2^l) = \max(e(p_1^r), \varepsilon(p_1^r, p_2^l))$ and repeat iteratively the process, finding the optimal value of p_2 for every possible choice of p_3 and so on. At the end, we will reach p_k establishing the optimal choice of p_{k-1} and then, by backpropagation, the optimal path from p_0 to p_k.

Now we want to study the computational complexity of this procedure; for this purpose consider that the most expensive operations are due not to the Viterbi algorithm but to the evaluations of the link costs $\varepsilon(p_i^r, p_{i+1}^l)$. Thus, we should reduce as much as possible these evaluations, and this can be done by considering some particular relationships between the costs of the links entering or leaving the same node. In other words, the minimization in (5) can be performed by considering only a subset of the p_i^r as candidates for being $a(p_{i+1}^l)$.

Proposition 7: For a fixed p_{i+1}^l, $e(p_{i+1}^l)$ in (5) (and thus $a(p_{i+1}^l)$) can be found with a binary search on p_i^r, evaluating only $\log_2 w_i$ link costs $\varepsilon(p_i^r, p_{i+1}^l)$.

Proof: First consider that $\varepsilon(p_i^r, p_{i+1}^a) \geq \varepsilon(p_i^r, p_{i+1}^b)$ if $a > b$ and this implies by induction (as it is obvious) that $e(p_i^a) \geq e(p_i^b)$ if $a > b$. Furthermore, if $a > b$, we clearly have $\varepsilon(p_i^a, p_{i+1}^l) \leq \varepsilon(p_i^b, p_{i+1}^l), \forall l = 1, \ldots, w_{i+1}$. This means that, for every fixed p_{i+1}^l, when r ranges from one to w_i, the values $e(p_i^r)$ are nondecreasing and the values $\varepsilon(p_i^r, p_{i+1}^l)$ are nonincreasing. Thus, suppose that for a given value of r, say, $r = a$, we have $e(p_i^a) > \varepsilon(p_i^a, p_{i+1}^l)$; then, in this point, clearly $\max(e(p_i^r), \varepsilon(p_i^r, p_{i+1}^l)) = e(p_i^r)$ and, in order to lower this value so as to find the minimum in (5), we must move p_i^r from p_i^a on the left. On the contrary, if for a value of r, say, $r = b$, we have $e(p_i^b) < \varepsilon(p_i^b, p_{i+1}^l)$, then we must move p_i^r on from p_i^b on the right, so as to decrease the value of $\varepsilon(p_i^r, p_{i+1}^l)$. The above argument implies that it is possible to search the optimum p_i^r with a binary search; we check first the point $p_i^{[w_i/2]}$, then $p_i^{[w_i/4]}$ or $p_i^{[3w_i/4]}$, depending on whether $e(p_i^{[w_i/2]}) > \varepsilon(p_i^{[w_i/2]}, p_{i+1}^l)$ or not, and so on, dividing by a factor of two the possible positions of p_i^r at every step, and thus finding the minimum in $\log_2 w_i$ steps. Note that if a value of r is found such that $e(p_i^r) = \varepsilon(p_i^r, p_{i+1}^l)$, then this r is optimal. ■

We now give an upper bound on the number of operations needed for the execution of Algorithm 6.

Proposition 8: Algorithm 6 needs at most an expected number of $O(n^2 \log n)$ operations, with n being the total number of samples.

Proof: We refer to Fig. 6 as a support for the computational complexity analysis. In the whole proof, we make use of the fact that for every i, we have $w_i \leq l_i$. We first consider the computations of the values $e(p_1^r)$ and $e(p_k)$, and then consider all other partition points. For finding the accumulated state metrics of the points p_1^r, we have to compute $\varepsilon(p_0, p_1^1), \varepsilon(p_0, p_1^2), \ldots, \varepsilon(p_0, p_1^{w_1})$, and thus w_1 approximations, each one of length less than or equal to l_1. So, the total number of expected operations needed for the p_1^r points is at most $O(w_1 \cdot l_1) \leq O(l_1^2)$. For finding $e(p_k)$, instead, from

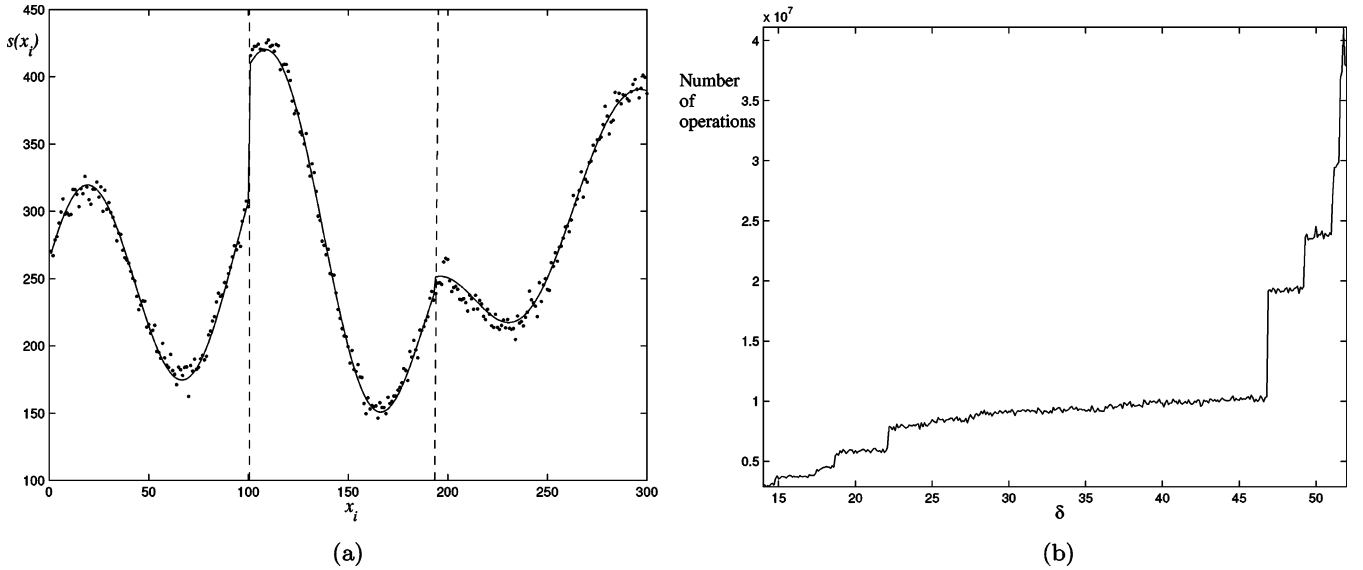


Fig. 7. Example of an optimal l^∞ signal approximation by means of mixed piecewise linear and cosinusoidal expansions. Given the dotted signal shown in (a), we have computed the optimal approximation obtained choosing a value of δ in the interval [14,52]. As we can see in (b), the number of operations increases with δ. In particular, it is interesting to see the increase in the value of the number of multiplications for some specific values of δ. For example, when δ reaches values near 47, we can see that the number of multiplications notably increases. This is due to the fact that when δ changes from 46.7 to 46.8, the point $p_1(\overleftarrow{g})$ goes from 100.5 to 94.5, thus with a consequent increase in the value of w_1 and then in the dimension of the trellis. (a) Signal approximation with $\delta \in [14, 52]$. (b) Number of executed operations when approximating the signal with respect to the value of $\delta \in [14, 52]$.

Proposition 7, we have to compute $\log_2 w_{k-1}$ approximations, each one of length less than or equal to $l_k + w_{k-1}$. So, for p_k, the expected number of operations is at most $O((w_{k-1} + l_k) \cdot \log_2 w_{k-1}) \leq O((l_{k-1} + l_k) \cdot \log_2 l_{k-1})$. For every remaining point[7] p_i, $1 < i < k$, we have to compute $w_i \log_2 w_{i-1}$ approximations of length less than or equal to $w_{i-1} + l_i$, and the expected number of operations is bounded by $O((l_{i-1} + l_i) \cdot l_i \cdot \log_2 l_{i-1})$. Thus the total number of expected operations is at most

$$O\left(l_1^2 + \sum_{i=2}^{k-1} ((l_{i-1}+l_i) l_i \log_2 l_{i-1}) + (l_{k-1}+l_k) \log_2 l_{k-1}\right) \tag{6}$$

and, considered that $l_i \leq n$ for all i (and thus that we can bound the logarithms with $\log_2 n$), we have at most

$$O\left(n \log_2 n \left(l_1 + \sum_{i=2}^{k-1} (l_{i-1} + l_i) + (l_{k-1} + l_k)\right)\right) \tag{7}$$

operations. Now, as $\sum_i l_i = n$, this quantity is at most $O(2n^2 \log_2 n) = O(n^2 \log n)$. ■

It is possible to show that this bound cannot be further lowered because we can construct an example of partition for which the algorithm has complexity of exactly $O(n^2 \log_2 n)$ expected operations. Nevertheless, this estimation can be very pessimistic in most practical situations. In many applications, in fact, we can suppose that the maximum length $\max_i(l_i)$ of the approximation intervals is asymptotically bounded[8] with respect to the number of points n. In this case we have the following.

Proposition 9: If $\max_i l_i$ is bounded with respect to the number of points n, then Algorithm 6 needs an expected number of $O(n)$ operations.

Proof: Suppose $l_i \leq L$, $\forall i$, independently of the value of n. Then (6) can be bounded by

$$O\left(L^2 + \sum_{i=2}^{k-1} 2L^2 \log_2 L + 2L \log_2 L\right) \leq O(kM) = O(k) \tag{8}$$

where $M = 2L^2 \log_2 L$. But, clearly, k satisfies $n/L \leq k \leq n$, and the complexity is thus $O(n)$. ■

Now that we have estimated the complexity of the method with respect to the number of points, it is important to clarify that the execution time is very much influenced by the selected error threshold. Even if at first glance this seems counterintuitive, we have to consider that the dimension of the obtained trellis depends on the error bound δ. Suppose, in fact, that the signal is such that it can be approximated with a number k of intervals if and only if the error threshold δ is in the interval $[\delta_1, \delta_2[$. Then, clearly, the obtained optimal solution has an error equal to δ_1 and does not depend on δ if it is in the considered interval. On the contrary, the two minimal approximations $\overrightarrow{g}$ and $\overleftarrow{g}$ depend on δ; and, in particular, the larger the value of δ, the more different their partition sets P. As a consequence, the constructed trellis varies from a trivial one for the value $\delta = \delta_1$ to a maximum dimension when δ approaches δ_2. This fact is better shown with an example; in Fig. 7(a), we can see the optimal approximation (solid line) obtained when approximating the given signal (dotted) with an error threshold in the interval [14,52], using as basis functions[9] $B = \{1, x, \sin(\pi x/50), \cos(\pi x/50)\}$. As we can see from Table I, this interval of values for δ leads to an optimal approximation that uses three intervals, (and has

[7]i.e., for evaluating the values $e(p_i^j)$, $j = 1, \ldots, w_i$, for a fixed $1 < i < k$.

[8]This assumption is completely natural when the variation of the number of samples n is due to the time windowing of a given signal. Consider, for example, an audio signal: unless we are talking of silence, it makes sense to suppose that the maximum interval length does not depend on the number of samples we are studying (if we are using a predefined constant sampling frequency).

[9]Note that it is often computationally useful, in practice, to use shift-invariant basis.

TABLE I
MINIMUM NUMBER OF INTERVALS AND OPTIMAL ERROR VALUE OBTAINED FOR VARIOUS ERROR THRESHOLD δ WHEN APPROXIMATING THE SIGNAL DOTTED IN FIG. 7(a)

δ	k	opt. er.	δ	k	opt. er.
120	1	101.94	11.55	8	11.29
101.93	2	52.72	11.28	9	10.68
52.71	3	13.91	10.67	10	10.60
13.90	4	13.74	10.59	11	10.42
13.73	5	13.05	10.41	12	10.04
13.04	6	11.58	10.03	13	9.93
11.57	7	11.56	9.92	14	9.43

a maximum error of 13.91). In Fig. 7(b), we can see how the number of operations used by the algorithm increases significantly with δ, as explained.

C. Perspectives for Representation of Signals by Irregular Samples

In the first section of this paper, we have recalled that every l^∞ (1-link) signal approximation problem in an m-dimensional linear space can be reduced to a linear program. Thus, it is not difficult to show that the solution of the problem leads to the identification of a (not necessarily unique) set of $m+1$ samples (which are in fact what we called *pivot points* in the straight line approximation case) that uniquely specify the optimal approximation. This means that there exists a set of $m+1$ samples (out of the, say, n) such that the optimal approximation is only due to them, and removing all the other $n-m-1$ samples does not change the optimality of this approximation. Thus, l^∞ approximation can be seen as a tool for irregular signal subsampling, in the sense that it automatically gives a subset of samples that bring the behavior of the whole signal (with a confidence related to the m and δ values). In the case of piecewise approximations, moreover, the study we have performed leads to the determination of a minimal number of points and a domain partition that optimally describe the whole signal. In Fig. 8, we show an example of the result of this subsampling procedure when applied to the electrocardiogram signal of Fig. 10 in the next section. It is clear, however, that the reconstruction of the approximation beyond the pivot points by using only these samples can be performed if the partition is known.

V. PIECEWISE LINEAR APPROXIMATIONS

In the preceding section, we have described an algorithm for finding the optimal piecewise approximation when working in general piecewise linear spaces G_T; in that case we considered that the approximation over every interval could be obtained by using the linear programming approach and thus with average time proportional to the number of samples. It is clear that if we are interested in piecewise linear approximations, the geometric method exposed in Section III-A must be preferred, as it gives much better performance.

Anyway, we show here that with a slight different version of the method exposed in Section III-A, it is possible to improve the construction algorithms for the minimal and optimal solutions. We have in fact the following result.

Proposition 10: Given n sample of a signal and an $l \leq n$, it is possible to find the optimal straight line approximations of the sets of points $S_j = \{s_i\}_{i=1,\ldots,j}$ with $j \leq l$ in at most $O(l)$ operations.

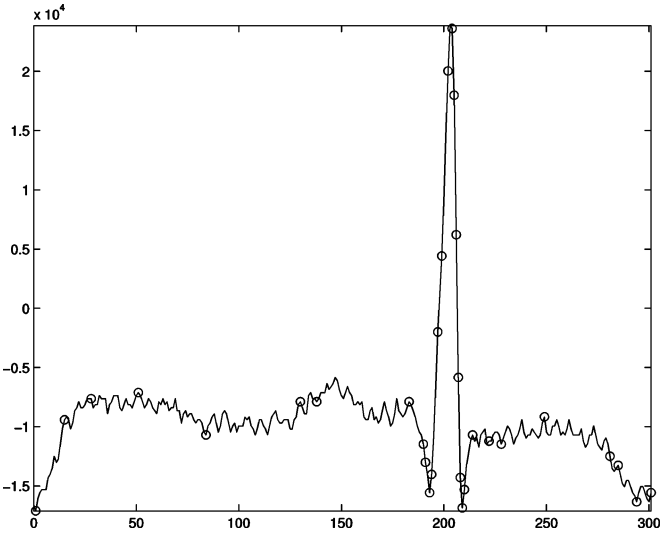


Fig. 8. Example of signal subsampling by means of l^∞ piecewise approximations. The circles represent the pivot points of the optimal segmentation subdomains. Here we have used the same electrocardiogram signal of Fig. 10, with the optimal approximation of Fig. 10(d).

Proof: The proof is constructive, in the sense that we show how to find the approximations of the sets S_j, $j = 1, \ldots, l$, in $O(l)$. Considering the approximation of S_l with the geometrical method of Section III-A, we note that the convex hull is constructed in a progressive way, i.e., by adding points from left to right and updating the polygon at every step. This means that before finding the convex hull of S_l we have found the convex hull of every S_j with $j < l$. Now, it is possible to see that the pivot points of S_j can be obtained from those of S_{j-1} in a number of operations O_j such that $\sum_j O_j = O(l)$.

Consider the convex hull Q_{j-1} with pivot vertices A_{j-1}, B_{j-1} and C_{j-1} and, for a generic point P of a convex hull Q, let $x(P)$ be the index m such that $P = s_m$. Consider now the new entering point s_j (see Fig. 9 for a graphical representation). We can distinguish three cases.

1) In the first case s_j lies in the strip of plane delimited by the lines passing through the pivot vertices of Q_{j-1}. In this case the pivot points do not change and thus $O_j = 0$.
2) In the second case s_j is outside the strip from the side determined by C_{j-1}. In this case s_j is the pivot B_j, A_j is the consecutive of s_j in Q_j, and C_j has to be searched to the right of B_{j-1} in $O(x(C_j) - x(B_{j-1}))$ operations.
3) In the third case s_j is outside the strip from the side determined by A_{j-1} and B_{j-1}. In this case s_j is still the pivot B_j, A_j is the vertex that precedes s_j in Q_j, and C_j has to be searched at the right of C_{j-1} in $O(x(C_j) - x(C_{j-1}))$ operations.

Considering that $x(C_{j-1}) < x(B_{j-1})$, we can see that the worst case is the third. Thus, in the worst case we need a number of operations that is $\sum_j O(x(C_j) - x(C_{j-1}))$ and, using the telescopic property, this is $O(x(C_l)) = O(l)$ operations. ■

As we have said, this fact is very useful in the study of piecewise linear approximations. In particular, we have the following result.

Proposition 11: For the case of piecewise linear approximations, Algorithms 5 and 6 require at most $O(n)$ and $O(n^2)$ expected operations, respectively.

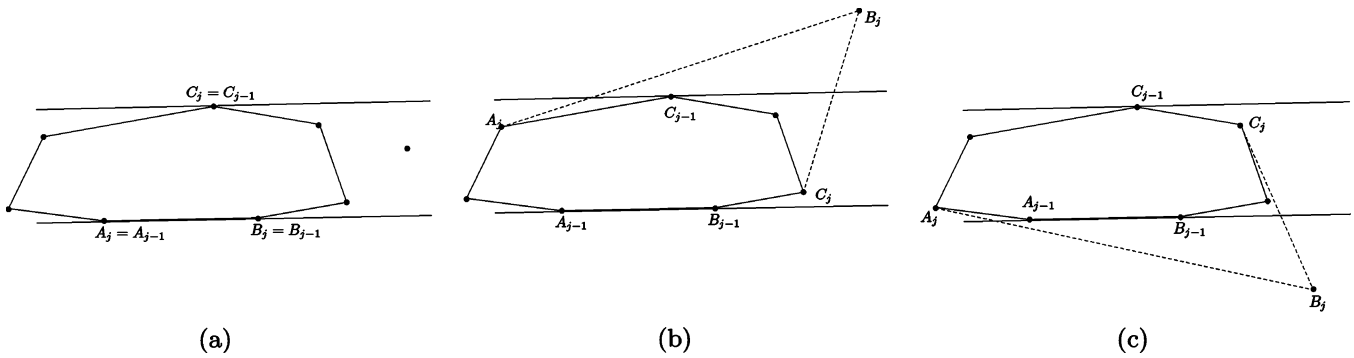


Fig. 9. Finding the pivot points A_j, B_j, and C_j of Q_j after the insertion of the new point s_j. We can distinguish three cases, every one being solvable with a number of operations that is at most $O(x(C_j) - x(C_{j-2}))$. (a) Case 1, (b) case 2, and (c) case 3.

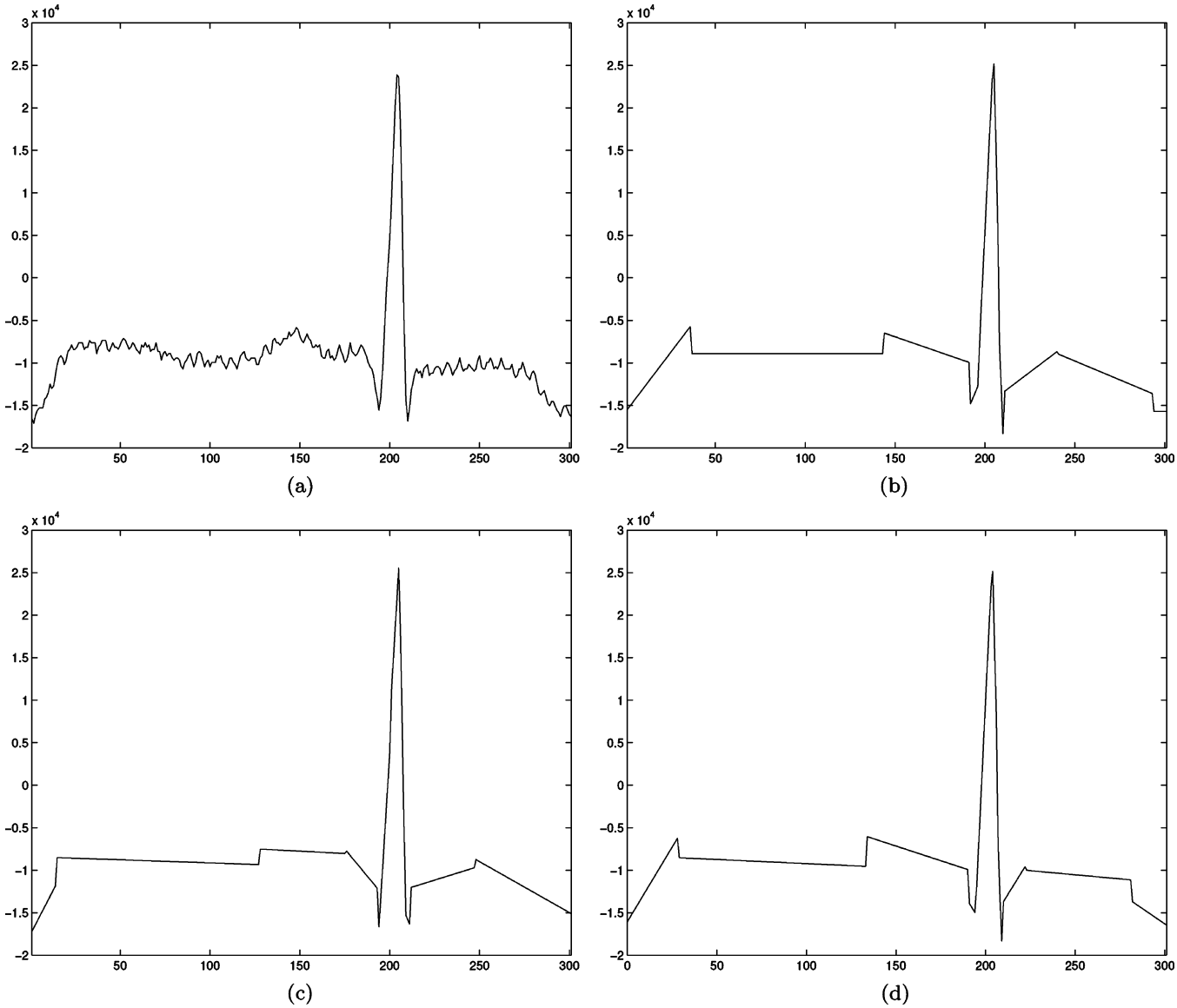


Fig. 10. Example of approximation of an electrocardiogram signal by means of piecewise straight line approximations. (a) Original signal, (b) left-to-right approximation ($\overrightarrow{g}$), (c) right-to-left approximation ($\overleftarrow{g}$), and (d) optimal approximation (f).

Proof: The result is essentially a consequence of the fact that, from Proposition 10, it is possible to avoid in Algorithms 5 (or better in Algorithm 4 used in Algorithm 5) and 6 the binary searches. In details, consider the construction of $\overrightarrow{g}$ (the same holds for $\overleftarrow{g}$). Using the progressive approximation construction explained above, we can scan the signal by adding a

TABLE II
PARTITION POINTS, ERRORS, AND NUMBER OF MULTIPLICATIONS FOR THE APPROXIMATIONS $\overrightarrow{g}$, $\overleftarrow{g}$, AND f, WHEN SETTING $\delta = 2000$ IN THE APPROXIMATION OF THE SIGNAL PLOTTED IN FIG. 10(a)

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	e	n. of operations
$\overrightarrow{g}$	36.5	143.5	191.5	196.5	204.5	207.5	210.5	240.5	293.5	1984	$6 \cdot 10^3$
$\overleftarrow{g}$	15.5	128.5	176.5	194.5	201.5	206.5	209.5	212.5	248.5	1941	$8 \cdot 10^3$
f	29.5	134.5	191.5	195.5	204.5	207.5	210.5	223.5	282.5	1631	10^5

point at every step, starting a new interval every time the error exceeds the given threshold δ. This way the number of operations for every interval $[p_{i-1}(\overrightarrow{g}), p_i(\overrightarrow{g})]$ is $O(l_i)$ and thus the total number of operations is $O(n)$.

In the same way, consider the evaluation of the generic $e(p_{i+1}^l)$ in the second step of Algorithm 6 (and refer to Fig. 6 for a graphical support). Instead of searching the minimum of $\max(e(p_i^r), \varepsilon(p_i^r, p_{i+1}^l))$ over r with a binary search (thus computing $\log_2 w_i$ approximations), we can find all the values $\varepsilon(p_i^r, p_{i+1}^l)$, $r = 1, \ldots, w_i$, in $O(w_i + l_{i+1})$ operations. In fact, we can first compute the approximation on $[p_i^{w_i}, p_{i+1}^l]$ and then add the points $p_i^{w_i-1}, p_i^{w_i-2}, \ldots, p_i^1$ on the left one by one, updating the convex hull and the optimal solution as explained above, for a total number of operations of $O(w_i + l_{i+1})$. This leads to a gain of a factor $\log w_i$ for every p_{i+1}^l and thus to a gain of $\log n$ in the complexity of the complete algorithm. ■

As an example of approximations by means of piecewise straight line functions, we show in Fig. 10 the results when applying the algorithm to an electrocardiogram signal. The signal samples are 16-bit signed integers (values from -32768 to 32767) and we have set an error threshold $\delta = 2000$. The algorithm has found a minimum necessary number of ten segments; in Table II we can see the values of the partition points for the minimal approximations $\overrightarrow{g}$ and $\overleftarrow{g}$ and for the optimal one f, together with the relative approximation errors and the associated computational complexity. Fig. 10 provides the original signal and its approximations.

VI. CONCLUSION

In this paper, we have presented new strategies for the approximation of signals in the l^∞ norm. We have shown a very efficient algorithm for finding the straight line approximation of a signal of n samples in $O(n)$ operation, leading to computation time that is up to eight times smaller than using the current linear programming techniques, and with very small memory usage. The extension of the theory of straight line approximation to higher dimensions (e.g., plane approximation for two-dimensional signals) has been so far experimentally validated with the use of the convex hull and the definition of pivot points. The theoretical formulations are still a matter of current research.

We have then studied the problem of piecewise signal approximations in linear spaces. Given an error threshold δ, we have shown how to find an approximation that uses a minimum number of intervals in $O(n \log n)$ operations in the worst case, which reduces to $O(n)$ for most practical situations. Furthermore, we have shown that for this minimum number of intervals, the optimal approximation can be found by using an l^∞ norm-based Viterbi algorithm in $O(n^2 \log n)$ operations in the worst case, which still reduces to be $O(n)$ in typical cases.

Finally, for the particular case of piecewise straight line approximations, we have shown that the worst case complexity of the two algorithms cited above can be reduced to $O(n)$ and $O(n^2)$, respectively, by using the geometric approach. The benefit of using the l^∞ norm has been finally discussed for the sake of discrete signal representation.

APPENDIX I

In this section, we prove the statements given in Section III-A on the geometrical properties of the convex hull Q of a set of points S. For better readability, we restate Lemmas 1–4 of Section III-A2, as they are also necessary for the proof of Proposition 1. We recall the used nomenclature. Given a set $S = \{s_i\}$ of n points in the plane, we call Q its convex hull. If k is the number of sides of Q, we call p_i, $i = 1, \ldots, k$, the vertices of Q in counterclockwise order, with p_1 the left-most one. For clarity, we add a point $p_{k+1} = p_1$ and set m, $m \leq k$, as the integer such that p_m is the rightmost vertex. For $i = 1, \ldots, k$, we call l_i the side $\overline{p_i p_{i+1}}$ and $v(l_i)$ the opposite vertex to the side l_i, i.e., the most distant vertex of Q from l_i in the direction orthogonal to l_i (distances between vertices and sides will always be considered in this sense in what follow). We say that $v(l_i)$ is x-internal to l_i if the vertical line through $v(l_i)$ cuts l_i.

Lemma 1: Every side of the lower hull has opposite vertex in the upper hull and *vice versa*.

Proof: This fact is somehow obvious. Anyway, consider a side l_i of the lower hull, and suppose p_m is more distant from l_i than p_1. This situation is shown in Fig. 11(a), where t' is the line parallel to line t to which l_i belongs. By the definition of m, $v(l_i)$ must lie at the left of p_m and, by the definition of opposite vertex, $v(l_i)$ must lie above line t'. Thus, it is easy to see that $v(l_i)$ must lie in the portion of the plane indicated with A. Any point p_j, $i < j < m$, of the lower hull must instead lie in the B area. Thus the point $v(l_i)$ belongs to the upper hull.[10] If p_1 is more distant from l_i than p_m, we obtain the equivalent symmetric situation shown in Fig. 11(b), which leads to the same conclusion. For the upper hull sides, we can operate symmetrically with a vertical flip and thus prove the converse. ■

Lemma 2: If we move from one side of the polygon to its consecutive in CCW direction, the respective opposite vertex, if it changes, moves in CCW direction too.

Proof: We consider the generic sides l_i and l_{i+1} with their opposite vertices as shown in Fig. 12; lines s and t are parallel to l_i and l_{i+1}, respectively. It is clear that $v(l_{i+1})$ cannot be farther than $v(l_i)$ from l_i and must be at least as distant as $v(l_i)$ from l_{i+1}. Thus $v(l_{i+1})$ must lie in the shaded portion of plane between s and t, and thus it is positioned in CCW direction with respect to $v(l_i)$. Note that this does not mean that $v(l_{i+1})$ is the consecutive vertex of $v(l_i)$ (see Lemma 4). ■

[10]Again remember that we consider p_1 and p_m to pertain to both upper and lower hull.

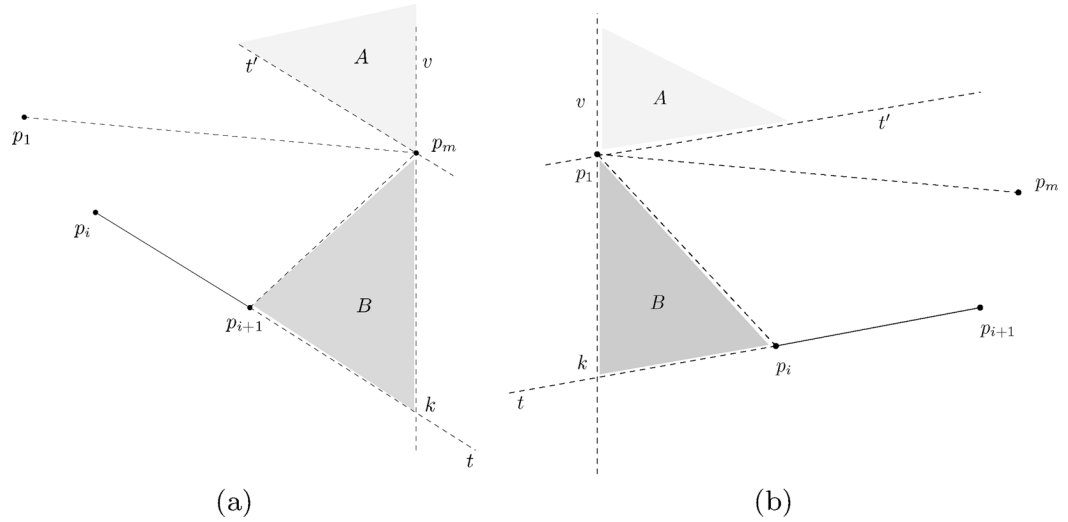


Fig. 11. Possible region for the opposite vertex of a lower hull side.

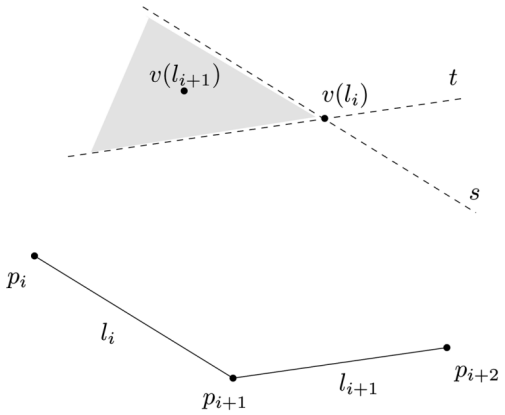


Fig. 12. Relation between the opposite vertices of two consecutive sides.

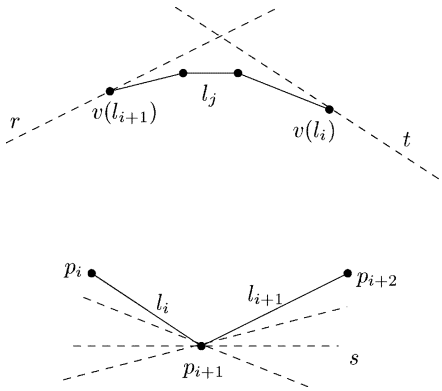


Fig. 14. Reciprocity property between sides and opposite vertices.

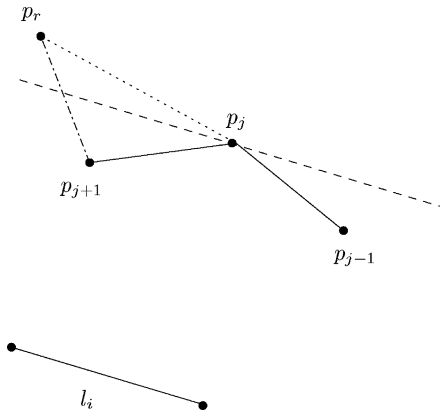


Fig. 13. The opposite vertex can be recognized considering only its neighbors.

Lemma 3: A vertex p_j, $1 < j \leq k$, is the opposite vertex of a side l_i, i.e., $p_j = v(l_i)$, if it is more distant from l_i than vertices p_{j-1} and p_{j+1}.

Proof: This follows directly by the convexity of the convex hull. Consider Fig. 13. If p_j is more distant than p_{j-1} and p_{j+1} from l_i, and if there were a vertex p_r more distant than p_j, then the segment $\overline{p_j p_r}$ would not be inside the convex hull, which is absurd. ■

Lemma 4: Given two consecutive sides l_i and l_{i+1}, their common vertex p_{i+1} is the opposite vertex of every side between $v(l_i)$ and $v(l_{i+1})$ (in the path not containing l_i and l_{i+1}, obviously).

Proof: Consider Fig. 14, where the position of the opposite vertices is justified and imposed by Lemma 2. Lines t and r are parallel to l_i and l_{i+1}, respectively. From the fact that $v(l_i)$ and $v(l_{i+1})$ are opposite vertices of l_i and l_{i+1} and from the convexity of the convex hull, we can see that every side between $v(l_i)$ and $v(l_{i+1})$ has a slope that is "intermediate" between the slopes of t and r. So, the generic side l_j between $v(l_i)$ and $v(l_{i+1})$ has a slope that is intermediate between those of l_i and l_{i+1}; this means that its parallel s through p_{i+1} leaves p_i and p_{i+1} in the same halfplane and thus p_{i+1} is more distant than p_i and p_{i+2} from l_j. From Lemma 3, this implies that p_{i+1} is the opposite vertex of l_j. ■

A. Proof of Proposition 1

We start by demonstrating that there exists at least one side l whose opposite vertex $v(l)$ is x-internal to it. Suppose that every side l_i has its opposite vertex $v(l_i)$ which is not x-internal; then, clearly, $v(l_1)$ must be on the right of l_1 and $v(l_{m-1})$ must be on the left of l_{m-1}. So, there must exist an integer $j < m$ such that

$v(l_{j-1})$ is on the right of l_{j-1} and $v(l_j)$ is on the left of l_j. Then, from Lemma 4, p_j is the opposite vertex to every side between $v(l_{j-1})$ and $v(l_j)$; the vertical line through p_j must cut one of these sides, and so there exists a side whose opposite vertex p_j is x-internal to it, so that the initial hypothesis was inconsistent.

Now, suppose we have three points A, B, and C of Q such that C is the x-internal opposite vertex to the side $\overline{AB}$. For these three points, the optimal linear approximation is easily proved to be the line r parallel to $\overline{AB}$ and equidistant from $\overline{AB}$ and C. The error produced by this line in approximating s at every x coordinate x_i is proportional to the distance of the point $s_i = (x_i, s(x_i))$ from the line; the way r has been selected[11] ensures that A, B, and C are the points of S most distant from r and so, the l^∞ approximation error of r is due to A, B, and C. But for these three points r is optimal, and so it is for the whole set S, as A, B, and C are peculiar vertices of the convex hull.

Finally, we show that there cannot exist another triplet of points A', B', and C' such that C' is x-internal to the side $\overline{A'B'}$. Supposing these three points exist, they should lead to an optimal solution r'. Calling $e(t; q_1, q_2, q_3)$ the error produced by the line t over the points q_1, q_2, and q_3, we should have

$$e(r'; A', B', C') \geq e(r'; A, B, C) \geq e(r; A, B, C) \quad (9)$$

since r' reaches its maximum error on A', B', and C', and r is optimum for A, B, and C. But symmetrically we have

$$e(r, A, B, C) \geq e(r; A', B', C') \geq e(r'; A', B', C'). \quad (10)$$

So the only possibility is that all these $\geq$ must be replaced by $=$ and, consequently, $r = r'$, which means that $\overline{AB}$ is parallel to $\overline{A'B'}$, contrarily to the initial hypothesis that Q has no parallel sides. This argument also proves that if Q has parallel sides,[12] the optimal solution is still unique, even if this is not true for the pivot points.

REFERENCES

[1] M. Powell, *Approximation Theory and Methods*. Cambridge, U.K.: Cambridge Univ. Press, 1981.

[2] G. A. Watson, "Approximation in normed linear spaces," *J. Comput. Appl. Math.*, vol. 121, pp. 1–36, 2000.

[11]Consider that all the points of S lie in the strip of plane between the line t passing through A and B and its parallel t' passing through C. The line r is exactly in the medium of this strip and the most distant points are those lying on t and t'.

[12]In this case it is necessary to adjust some technical details such as the definition of *opposite vertex,* but the main arguments and their consequences still hold.

[3] E. Meijering, "A chronology of interpolation: From ancient astronomy to modern signal and image processing," *Proc. IEEE*, vol. 90, no. 3, pp. 319–342, Mar. 2002.

[4] D. G. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison Wesley, 1984.

[5] F. Preparata and M. Shamos, *Computational Geometry: An Introduction*. Berlin, Germany: Springer-Verlag, 1985.

[6] J. E. Goodman and J. O'Roarke, *Handbook of Discrete and Computational Geometry*. Boca Raton, FL: CRC Press, 1997.

[7] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[8] I. Barrodale and C. Phillips, "Solution of an overdetermined system of linear equations in the Chebyshev norm [F4] (algorithm 495)," *ACM Trans. Math. Software*, vol. 1, no. 3, pp. 264–270, 1975.

[9] R. Seidel, "Small-dimensional linear programming and convex hulls made easy," *Discrete Comput. Geom.*, vol. 6, pp. 593–613, 1991.

[10] N. Megiddo, "Linear programming in linear time when the dimension is fixed," *J. ACM*, vol. 12, pp. 114–127, 1984.

[11] M. Rajeev and R. Prabhakar, *Randomized Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.

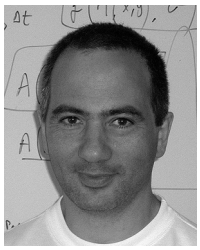
[12] M. Dalai and R. Leonardi, "l^∞ norm based second generation image coding," in *Proc. ICIP'04*, Oct. 2004, pp. 3193–3196.

[13] D. Avis and D. Bremner, "How good are convex hull algorithms?," in *Symp. Comput. Geom.*, 1995.

[14] R. Graham, "An efficient algorithm for determining the convex hull of a finite point set," *Inf. Proc. Lett.*, vol. 1, pp. 132–133, 1972.

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

Marco Dalai (S'05) was born in Manerbio (BS), Italy, in 1979. He received the Dr.Eng. degree in electronic engineering from the University of Brescia, Italy, in 2003, where he is currently pursuing the Ph.D. degree in information engineering at the Department of Electronics for Automation.



Riccardo Leonardi (S'80–M'87) received the diploma and Ph.D. degrees in electrical engineering from the Swiss Federal Institute of Technology, Lausanne, in 1984 and 1987, respectively.

After one year as a Postdoctoral Fellow with the University of California at Santa Barbara, he spent three years with AT&T Bell Laboratories as a Member of Technical Staff, performing research activities on visual communication. In 1992, he joined the University of Brescia, Italy, to lead research and teaching in the field of telecommunication. He holds the Signal Processing Chair. His main research interests cover the field of digital signal processing applications, with a specific expertise on visual communications, and content-based analysis of audio-visual information. He has published more than 100 papers on these topics. Since 1997, he has acted as an Evaluator and Auditor of several European Commission RTD programs.