Optimal piecewise linear image coding

Dietmar Saupe

Institut für Informatik, Universität Freiburg
Am Flughafen 17, 79110 Freiburg Germany

ABSTRACT

Piecewise linear (PL) image coding proceeds in three steps: 1. A digital image is converted into a 1D-signal using a scanning procedure, for example by scanning lines in a zig-zag or Hilbert order. 2. The signal is approximated by the graph of a piecewise linear function, which consists of a finite number of connected line segments. 3. Entropy encoding of the sequence of the segment end points. In this step differential coding can be used for one or both coordinate sequences of the end points. To ensure a desired approximation quality a constraint is imposed, e.g., on the root-mean-square error of the PL signal. In this paper we consider uniform approximation (L_∞-distortion limited compression). Two problems are addressed: First, an optimal PL approximation in the sense of a minimal number of segments is to be obtained. Second, when entropy coding of the segments is used, how can one jointly optimize the variable length code and the PL approximation yielding a better or even minimal rate without violating the uniform error bound? The first problem is solved by dynamic programming, the second is approached by using Huffman coding and an annealing procedure in which the design of the Huffman tables and the dynamic programming is alternatingly iterated using a cost function that reflects the codeword lengths of the current variable length code. This algorithm is guaranteed to converge to a (locally) minimum length code. We describe the algorithms, implementation issues, compare two different scanning procedures, the zig-zag line scan and the Hilbert scan, and report results for encoding various test images.

Keywords: nearlossless image coding, sequence coding, piecewise linear coding, fan-based coding, dynamic programming, distortion limited compression, annealing, Hilbert scan.

1. INTRODUCTION

In many applications, for example medical imagery, SAR satellite pictures, or numerical weather simulations, the large amount of data to be stored or transmitted asks for data compression. Since lossless coding usually gives a compression ratio of at most 2:1, lossy coding methods have to be employed when higher ratios are needed. In some cases the sensitivity of the data does not allow a lossy method to be evaluated by a global error measure like mean square error or peak signal to noise ratio. Instead, there may be a need for a guarantee that the intensity at a pixel has not been changed by more than a certain tolerance. Thus, the error in each pixel has to be controlled, leading to L_∞-distortion limited compression. There are several approaches for ensuring a uniform error bound. A common technique works with entropy coded DPCM and quantization. By distorting the original intensity values within the error bounds one may minimize the entropy of the (modified) signal[1] or minimize the entropy of the quantized prediction error sequence.[2] Both methods employ dynamic programming techniques. Also distortion limited vector quantization[3] has been considered. An alternative is to apply any lossy coding scheme to the signal and then to encode the residual error image in the prescribed L_∞-distortion limited sense. The decoder, receiving both codes, first reconstructs the first, lossy version, and then adds the near-losslessly coded residual to obtain a near-lossless version of the original.

In this paper we discuss piecewise linear coding of the one-dimensional signal, special cases of which are also known as fan based coding (see Section 2) since the 1960's. This method regards an image as a one-dimensional signal, which can always be achieved using a line scan, or, e.g., a Hilbert scan[4] of the image. Whether such a restriction is a suitable approach to image coding is debatable, but this question is not addressed in this paper.

Piecewise linear coding is a generalization of run-length-encoding where a signal is split up into segments that can each be approximated by a linear function. A segment is coded by its length and the increase or decrease in signal value along the segment. At the start, also the initial signal value of the first segment has to be coded. For example, the signal (3,3,2,1,3,5,7,5,4) can be represented (in this case, without loss) as 3(1,0)(2,-1)(3,2)(3,-1). If during the decoding one of the reconstruction signal values is not an integer, rounding is performed.

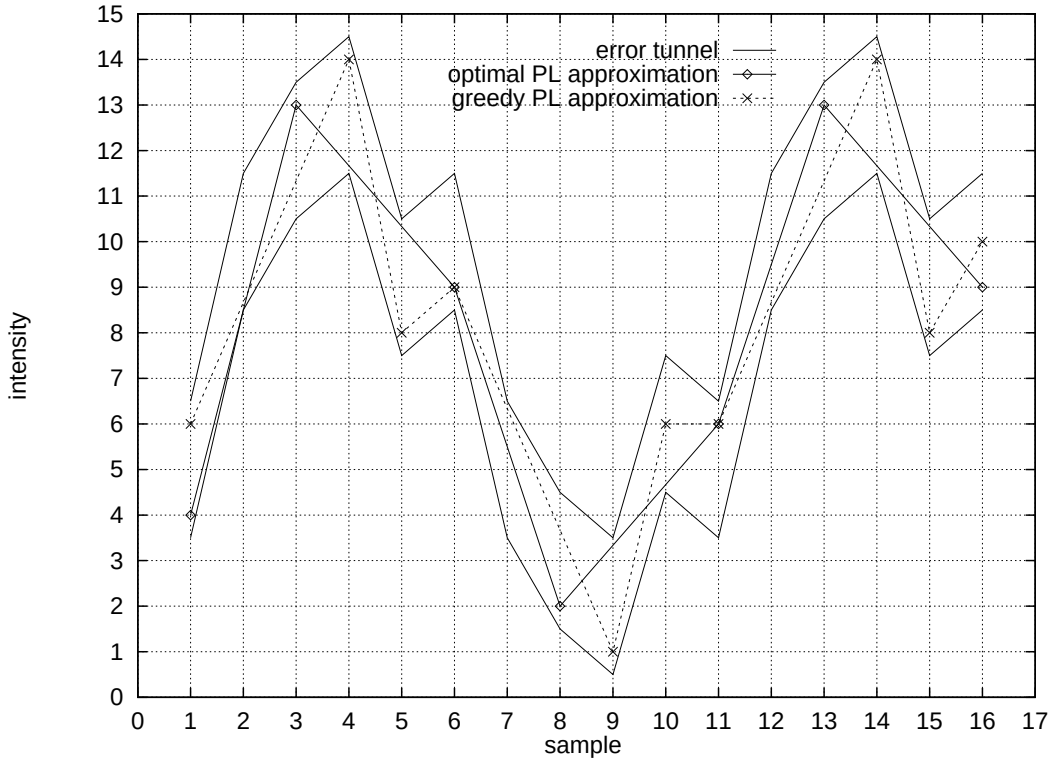


Figure 1. Optimal and greedy PL approximation of a short signal consisting of the sequence of intensity values (5,10,12,13,9,10,5,3,2,6,5,10,12,13,9,10). The maximum error allowed at a sample point is $t = 1$.

The optimization problem is to compute a piecewise linear approximation to the original signal within the distortion bound t such that the number of runs in the run-length coding described above is minimal. In Ref. 5–7 and many other papers a suboptimal greedy method that works in linear time is proposed. Essentially, it proceeds as follows. The image is transformed into a 1-dimensional signal, e.g., by a Hilbert scan. Then the linear segments are successively generated: starting at the endpoint of the last determined segment, the new segment is chosen to be the one of greatest possible length. In addition to the run-length coder, an entropy coder can be used to further improve the compression as a postprocess.

Figure 1 provides an example for a signal and associated optimal and greedy piecewise linear approximations for a uniform error bound of $t = 1$. Note that the PL approximations are constrained to an *error tunnel* which is obtained by translating the original signal up and down by $t + 1/2$ intensity units. The optimal PL approximation requires 6 segments, to be compared with 9 segments for the greedy version. The slope of the PL curve changes at segment end points, which we also call *break points*.

In the following section we review the literature on piecewise linear approximation that is most relevant for this work. In Section 3 we introduce the dynamical programming algorithm for minimizing the number of segments for the case of L_∞-distortion limited piecewise linear approximation with the constraint that the break points are on the integer grid. Section 4 discusses important implementation issues. In the following Section 5 we present an iterative algorithm aimed at finding shortest possible codes for PL approximations that satisfy the uniform error bound. In it we use Huffman coding and an annealing procedure in which the design of the Huffman tables and the dynamic programming is alternatingly iterated using a cost function that reflects the codeword lengths of the current variable length code. This algorithm is guaranteed to converge to a (locally) minimum length code. In Section 6 we report results for encoding various test images.

It is not the purpose of this research to compare the method of piecewise linear image coding with other competing techniques for L_∞-distortion limited image compression (although we give one example near the end to demonstrate

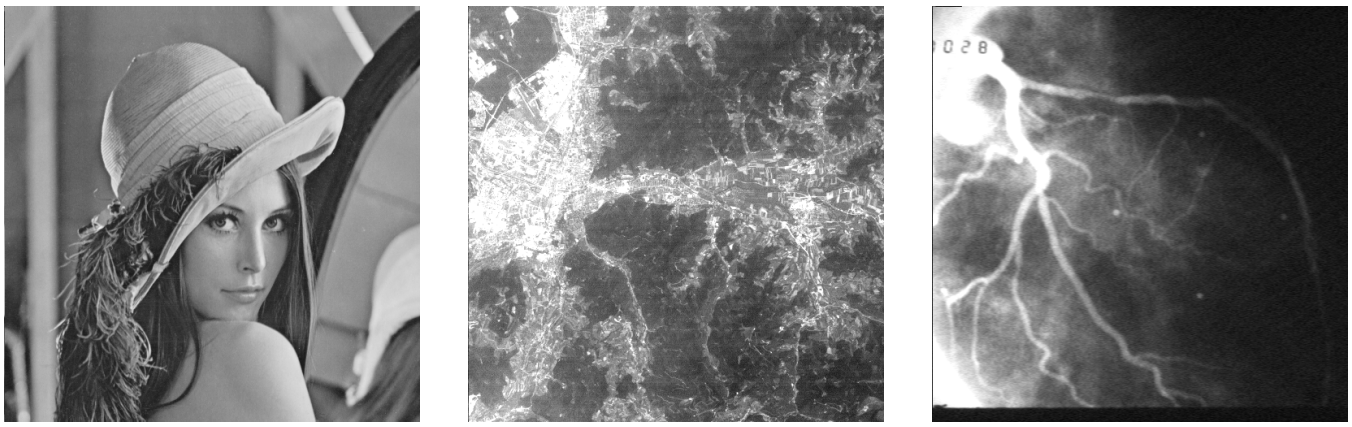


Figure 2. These three test images, called Lenna, Landsat, and Angio are 8 bpp grey scale images of resolution 512×512 pixels. The Landsat satellite image is of a part of the Black Forest in Germany with the city of Freiburg located left of the image center.

the limitations of PL image coding). We have performed such a study,[8] where, however, the annealing procedure was not tested.

2. PREVIOUS WORK

There exists a large body of literature on piecewise linear approximation to curves which is scattered over several fields:

- mathematics (approximation theory, numerical mathematics, operations research),
- computer science (computational geometry, computer graphics, pattern recognition),
- engineering (signal processing, data compression), and
- application areas such as biomedicine and aeronautical science.

A lot of that started in the 1960's. It is impossible now, to have a complete overview, and, thus, some of the basic techniques have been reinvented several times over the years. Work on PL approximation of curves can be classified according to several criteria:*

1. type of norm used for evaluating the quality of approximation (most commonly used are the L_2-norm and the maximum norm (L_∞),
2. with or without constraints on the placement of break points (free in the plane, or constrained to a submanifold like the given curve, or restricted to a grid),
3. continuity type (segment are required to be connected yielding a continuous PL curve, or not, giving a collection of disconnected segments in the plane, approximating the given curve),
4. the given curve is coming from discrete data (for example, measurement points connected by linear segments, yielding a piecewise linear curve), or from a continuous model such as a closed form formula,
5. algorithms are greedy, yielding a PL approximation with a small but not minimal number of segments, or optimal, requiring more computation time,
6. methods designed for general curves in the plane or specifically for curves that are graphs of functions.

*This list is inspired by that in Ref. 9, which also contains more references and a broader review of the work prior to 1986.

In this paper we are interested only in methods that are designed for curves, that can be seen as graphs of functions, rather than general curves in the plane.

There are a lot of contributions for so called *fan based algorithms*. They take as input a sequence of points to be approximated using a uniform error constraint. End points of line segments are required to be from the original data, thus, this is a subsampling scheme. Assume that the PL approximation has been generated up to a current data point, which we call the left node. Then one defines as the right node the next data point farthest down the list, such that all data points in between the left and right node are within a distance of at most the given tolerance from the connecting line segment between the left and right node points. A natural way to compute the right node is to use a cone shaped sector with its peak at the left node, opening up to the right so that the next feasible line segment is constrained to be within this sector. Each time a point is processed, the sides of the sector may have to be adjusted appropriately (reducing the angle of the sector). The first time a data point is encountered which is outside of the sector, the right node must be chosen to be the last one within the sector. The attribute "fan based" stems from the geometrical shape of the region used in this method. See Refs. 6,7,5,10 and references contained in them.

The fan based algorithm as described above is online and greedy. An optimal algorithm, producing the minimal number of segments required to satisfy the uniform approximation criterion was given by Dunham.[9] In this approach the PL approximation is required to have its break points on the given data curve to be approximated, the approximations are continuous, and the error is uniformly bounded. However, it does not seem natural to force zero error at the break points while allowing some fixed error elsewhere. In our work here we extend Dunham's methods in this respect, allowing a maximum error at all sites.

In Dunham's work as well as our's the principle of optimality from dynamic programming is used to obtain optimal solutions. The first contribution that recognized the potential of dynamic programming for the problem of optimal piecewise linear approximation was given by Bellman[11] in 1961. Bellman's work was continued by Gluss.[12] These works are different from Dunham's and our's, as they address the best approximation problem for curves, where the number of segments in the PL approximation is given as an input parameter and held constant (thus, optimality is with respect to approximation error), and, moreover, the PL approximation need not be continuous.

Now, again in the case where the given data are discrete points in the plane, let us consider that no constraints on the break points are set except that the segments of the PL approximation are required to connect up yielding a continuous piecewise linear curve. Imai and Iri[13] give an algorithm minimizing the number of segments using methods from computational geometry (efficient computation of visibility polygons). The unconstrained break points in the PL approximation in general will have real-valued components, which are not suitable for efficient coding. Thus, Baskaran et al[6] add a quantization step which forces the break points that are output by the Imai-Iri algorithm to lie on an integer grid. This postprocessing may potentially destroy the required approximation property, however, only by a small amount.

We close this section with a note about efficient encoding. In a 1962 paper of Gluss[12] entitled "A line segment curve-fitting algorithm related to optimal encoding of information" the amount of storage for the PL approximation is taken to be proportional to the number of line segments (obtained by fixed-length encoding). In this case minimizing the number of segments is equivalent to minimizing the bit rate. In all of the rest of the literature we have seen, the encoding of the PL approximation was not explicitly considered, or if it was, then encoding took place as a postprocess and was not integrated as a part of the optimization procedure. Our work here also fills this gap.

3. OPTIMUM PL APPROXIMATION BY DYNAMIC PROGRAMMING

Minimizing the number of runs among all PL approximations to a given set of data within the uniform error bound t can be solved via dynamic programming. To achieve this one may represent each sample of the data by $2t+1$ nodes in a labeled directed acyclic graph, corresponding to all its possible intensity reconstruction levels. See Figure 3 for an illustration. The edges in the graph correspond to admissible runs, connecting two samples with the graph of a linear function of the sample position. The labels of the edges are all identical to 1, providing a measure of the number of segments in a run-length coded signal. Now the problem of finding the optimal run-length code for the signal is equivalent to finding the shortest path starting at one of the $2t+1$ reconstruction levels of the first sample and ending at one of the levels for the last sample. It is well known that such problems can be solved by dynamic programming.

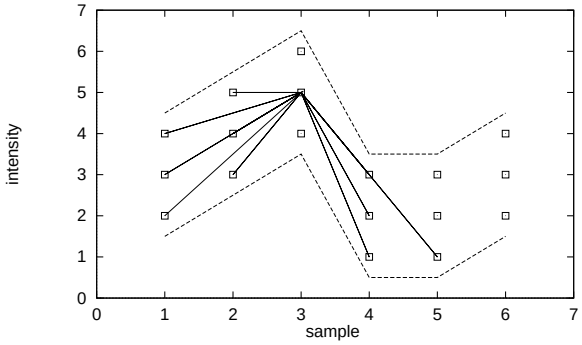


Figure 3. Illustration of a part of the directed acyclic graph for shortest path computation by dynamic programming. The given signal is (3,4,5,2,2,3) and the uniform error bound is $t = 1$. The nodes of the graph are shown as small squares, each one corresponding to an admissible reconstruction value for a signal sample. Two nodes are linked by an edge, if the line segment that connects them lies within the error tunnel outlined by the dotted boundaries. To avoid clutter only those edges are shown, that connect to the node at sample 3 and intensity 5. All edges are oriented from left to right. In the figure the lines shown simultaneously display linear signal approximations between the nodes connected by edges.

For a more formal statement of the problem and its dynamic programming solution, let us begin by introducing the following notation for piecewise linear interpolation and approximation.

Definition 1. *Let $P_i = (x_i, y_i), i = 1, ..., N$ with $x_1 < x_2 < \cdots < x_N$ denote N points in the plane with integer coordinates. The corresponding* piecewise linear (PL) interpolation function $\varphi : [x_1, x_N] \to \mathbf{R}$ *is obtained by setting*

$$\varphi(x) = \varphi(x; x_1, y_1, ..., x_N, y_N) = y_i + \frac{x - x_i}{x_{i+1} - x_i}(y_{i+1} - y_i), \quad x \in [x_i, x_{i+1}], \quad i = 1, ..., N-1.$$

A function $\tilde{f} : [x_1, x_N] \to \mathbf{R}$ is called a PL-approximation candidate function for the data $(P_i)_{i=1,...,N}$ with error bound $t \in \{0, 1, 2, ...\}$, *if there is a subsequence of K integers $(i_j)_{j=1,...,K}$ with $1 = i_1 < i_2 < \cdots < i_K = N$, $2 \le K \le N$, and integers*

$$\tilde{y}_{i_j} \in \{y_{i_j}, y_{i_j} \pm 1, ..., y_{i_j} \pm t\}, \quad j = 1, ..., K$$

such that $\tilde{f}$ is the corresponding PL interpolation function

$$\tilde{f}(x) = \varphi(x; x_{i_1}, \tilde{y}_{i_1}, ..., x_{i_K}, \tilde{y}_{i_K}).$$

The candidate function $\tilde{f}$ is called a PL approximation of the data $(P_i)_{i=1,...,N}$ with error bound $t \in \{0, 1, 2, ...\}$, *if, in addition, the function $\tilde{f}$ satisfies*

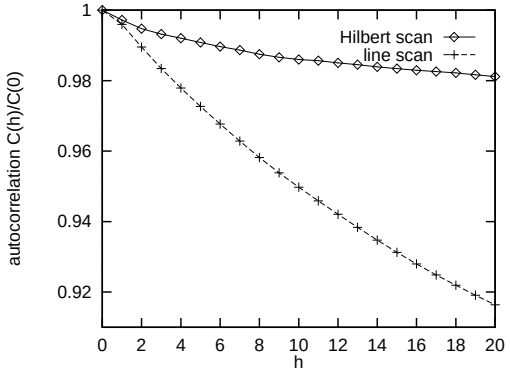
$$y_i - t - \frac{1}{2} \le \tilde{f}(x_i) < y_i + t + \frac{1}{2}, \quad i = 1, ..., N \tag{1}$$

Given a PL approximation of the original data $(P_i)_{i=1,...,N}$ with error bound t as output of some PL encoder, then the decoder would produce integer valued reconstruction values $\hat{y}_i, i = 1, ..., N$ by rounding, $\hat{y}_i = \text{round}(\tilde{f}(x_i))$. Thus, equation (1) states that for all $i = 1, ..., N$ the uniform error bound is observed, i.e., $|\hat{y}_i - y_i| \le t$.

The number of PL approximation candidate functions is large. For data of size N and error bound t, the total number is

$$\sum_{K=2}^{N} \binom{N-2}{K-2} (2t+1)^K. \tag{2}$$

The above definitions are set up for the case of discrete data with integer coordinates, because this seems to be the most common case for signal encoding in practice. It is easily modified to apply to suit a more general setting. The



	Line Scan		Hilbert Scan	
t	# segments	rate bpp	# segments	rate bpp
0	232846	5.17	233870	4.99
1	153639	4.14	157635	4.02
2	107704	3.33	112726	3.24
3	79124	2.71	83373	2.62
4	61703	2.26	63797	2.14
5	50869	1.95	50811	1.80
10	28533	1.21	23466	0.94
15	20361	0.90	14178	0.61

Figure 4. Plot of the normalized autocorrelation function $C(h)/C(0)$ with $C(h) = \sum_i S(i)S(i+h)$ for the scans S of the 512×512 test image Lenna. The table on the right shows the statistics of the resulting optimal piecewise linear approximations for the line scan and the Hilbert scan for different uniform error bounds t. Although the Hilbert scan exhibits a stronger autocorrelation, the ordinary line scan requires fewer line segments for PL approximations, except for larger error bounds.

objective of optimum piecewise linear approximation of discrete integer-valued signals with uniform error bound t can now be stated as follows.

Problem 1. *Let $P_i = (x_i, y_i), i = 1, ..., N$ with $x_1 < x_2 < \cdots < x_N$ be N points in Z^2, where Z is the set of integers. Moreover, let $t \in \{0, 1, 2, ...\}$ be a given error bound. Then the* Problem of optimal PL approximation *is to find a PL approximation $\tilde{f}(x) = \varphi(x; x_{i_1}, \tilde{y}_{i_1}, ..., x_{i_K}, \tilde{y}_{i_K})$ of the data with error bound t such that its size K is minimal.*

This problem can be solved as follows. Let $F_t(m, s)$ for $m = 1, ..., N$ and $s = 0, \pm 1, ..., \pm t$ be the function denoting the minimal size K of a partial PL approximation ending in the point $(x_m, y_m + s)$. More precisely, $F_t(m, s)$ is the minimal size K of all PL approximations $\tilde{f}(x) = \varphi(x; x_{i_1}, \tilde{y}_{i_1}, ..., x_{i_K}, \tilde{y}_{i_K})$ for the data $(P_i)_{i=1,...,m}$ such that $x_{i_K} = x_m$ and $\tilde{y}_{i_K} = y_{i_K} + s$. Then a solution of the optimization problem will give the minimal size $F_t^*(N)$, where

$$F_t^*(m) = \min_{s=0,\pm 1,...,\pm t} F_t(m, s), \quad m = 1, ..., N. \tag{3}$$

Moreover, the following recursion formula holds,

$$\begin{aligned} F_t(m, s) &= 1 + \min_{(i,r) \in I_{m,s}} F_t(i, r) \\ F_t(1, s) &= 0 \end{aligned} \tag{4}$$

for $m = 2, ..., N$ and $s = 0, \pm 1, ..., \pm t$ where the index set $I_{m,s}$ is defined as

$$\begin{aligned} I_{m,s} = \{(i, r) \quad | \quad & i < m, r = 0, \pm 1, ..., \pm t \text{ such that } \varphi(x; x_i, y_i + r, x_m, y_m + s) \\ & \text{is a PL approximation of } (P_i, P_{i+1}, ..., P_m) \text{ with error bound } t\} \end{aligned} \tag{5}$$

This is just a consequence of the principle of optimality. We remark that due to equation (4) we immediately obtain $F_t(2, s) = 1$ for all $s = 0, \pm 1, ..., \pm t$.

Also note that the case considered by Dunham[9] (break points on original data points) is a special case of the above. To see that, simply define $F_t^*(m) = F_t(m, 0)$ and replace in equation (4) both s and r by 0. This also simplifies the definition of the index set $I_{m,0}$.

The optimization problem discussed here is different from the more common problem of finding the shortest path in a trellis. In a trellis a node at some sample location is connected only to nodes corresponding to the neighboring samples on the left and right, which gives a total of $(2t + 1)^N$ paths. Here a feasible path may connect nodes from levels that are far from each other, yielding the maximal number of paths given in the expression in (2). Therefore, the complexity of the optimization problem is much greater in our case.

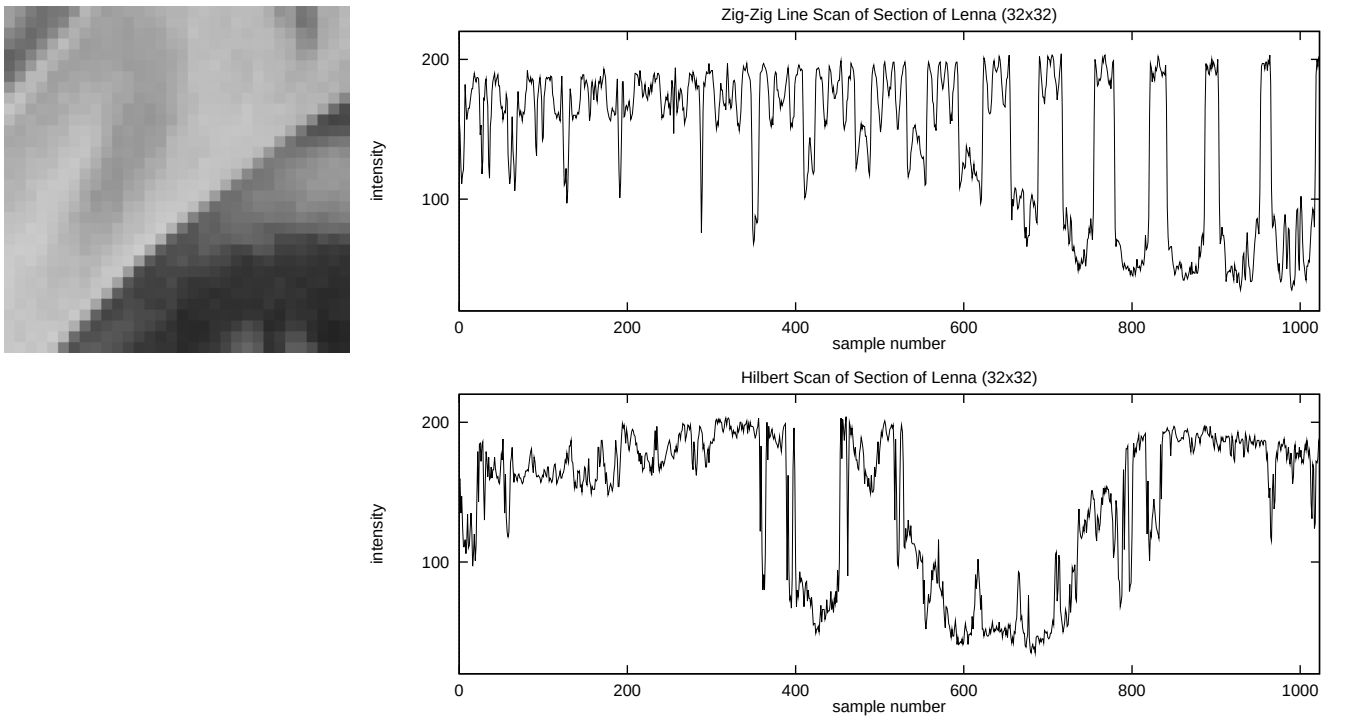


Figure 5. A section of 32×32 pixels of the test image Lenna (the bright band is a part of the hat).

The computation of $F_t^*(N)$ seems straightforward. For $m = 3, ..., N$ (in this order) and each $s = 0, \pm 1, ..., \pm t$ evaluate formula (4). Finally, use (3) to compute the minimal size $F_t^*(N)$. In order to also retrieve the optimal PL approximation with its break points $(x_{i_k}, \tilde{y}_{i_k})$ we need to store for each (m, s) not only the value of $F_t(m, s)$, but also one of the pairs (i, r) that led to the minimum in (4), i.e., from the set $\arg\min_{(i,r)\in I_{m,s}} F(i, r)$. In practice, this straightforward approach fails for complexity reasons. The number of checks in (5) whether $\varphi(x, x_i, y_i + r, x_m, y_m + s)$ is a PL approximation of $(P_1, ..., P_m)$ with error bound t is just too large, namely $(2t + 1)^2 N(N - 1)/2$. Clearly, a more efficient implementation is needed.

4. IMPLEMENTATION

In this section we give a brief description of an efficient implementation of the computation of optimal PL approximations by dynamic programming. It is motivated by the underlying directed acyclic graph (DAG). As pointed out, a brute force approach fails due to the complexity. The DAG should be kept as small as possible by pruning those parts that provably do not appear in the shortest path. In this way memory can be saved avoiding swapping of disk space. Moreover, data structures should be provided which help to avoid unnecessary calculations.

According to equation (4) we need to consider nodes for all tuples (m, s) with $m = 1, ..., N$ and $|s| \le t$. For each node we define a (forward) horizon as follows.

$$\begin{aligned} \text{horizon}(m, s) = \max\{m' \quad | \quad & m' \le N \text{ and } \varphi(x; x_m, y_m + s, x_{m'}, y_{m'} + s') \text{ is a PL approximation of} \\ & (P_m, P_{m+1}, ..., P_{m'}) \text{ with error bound } t \text{ for some } s' = 0, \pm 1, ..., \pm t\} \end{aligned}$$

In words, the horizon at a node is given by the index m' of the x-coordinate of the farthest point $(x_{m'}, y_{m'} + s')$ that can been "seen" from the point $(x_m, y_m + s)$ in the error tunnel. The horizon can be computed with the fan based method using only integer arithmetic. For each node a structure with the following fields is maintained:

- the x-coordinate x_m,
- the intensity difference w.r.t. y_m, i.e., $s = \tilde{y}_m - y_m$ with $|s| \le t$,

- a pointer to the last node on the optimal path leading to the current node (m, s), we call this last node the *parent node* of (m, s),
- an auxiliary pointer to a next node,
- the number of child nodes to which this node is a parent node.

There are two types of nodes, active and passive nodes (explained below). Passive nodes do not require to be organized in a special form, while active nodes are kept in a collection of node lists, each implemented as a linked list. All nodes (m, s) in a given node list share the same minimal number of segments, $F_t(m, s)$, of a shortest path leading to (m, s). The following fields are given in a node list:

- the common value of $F_t(m, s)$ of all nodes (m, s) in the node list,
- a pointer to the first node in the node list (the following nodes can be retrieved by recursively dereferencing the "next node" pointer given in each node),
- the maximum of horizon(m, s) for all nodes (m, s) in the node list,
- the minimum and the maximum of the x-coordinates x_m for all nodes (m, s) in the node list,
- a pointer to the next node list, for which the common value of $F_t(m, s)$ will be increased by 1.

The nodes in each node list are sorted with decreasing x-coordinate values. The linked list of node lists can be referenced by a pointer to the first node list, which is characterized by the smallest number $F_t(m, s)$.

A node kept in memory can have one of three states. The *current* node is the one under investigation. After allocating storage space its fields are computed and set. The search for the parent node only needs to consider the *active* nodes. A node is active, if all its fields have been set and provided that it cannot be excluded from further consideration. An entire node list will be excluded when its maximum horizon is smaller than the x-coordinate of the current node. An excluded (formerly active) node will become *passive* if it is the parent to some other active node (i.e., when the field of its number of child nodes is positive). Otherwise, the previously active node can be deleted. The deletion of a node must be carried out recursively, since its parent node should also be deleted, if the parent node has just this one child node. The passive nodes are no longer kept in lists of nodes, but they simply form an unorganized set of nodes.

For a given current node at (m, s) the search for its parent node proceeds as follows. The levels $m-1, m-2, m-3, ...$ are considered sequentially. In each such level $m - k$ we compute the minimal and maximal y-values that are accessible from the current node. More precisely, we compute $r_{\min}$ and $r_{\max}$ such that $(m - k, r) \in I_{m,s}$ if and only if $r \in \{r_{\min}, ..., r_{\max}\}$. Then the node lists are checked for containment of such a node $(m - k, r)$, beginning with the "first" list, i.e., the list with associated minimal number F_t. An entire node list can be disregarded, if the range from its minimal x-coordinate to its maximal one does not contain $m - k$. The rest of the procedure is straightforward taking advantage of the sortedness of the node lists. After the optimal parent node has been identified the current node (m, r) can be inserted into its proper position of the node list that corresponds to the value of $F_t(m, r)$. If a node list for the given value of $F_t(m, r)$ does not yet exist, a new one will be created and appended to the list of node lists. The horizon(m, r) needs to be calculated, and the fields in the corresponding node list are properly updated.

A worthwhile speedup of this scheme can be obtained by observing that often several consecutive nodes in a current level have the same parent node in the "first" node list. Thus, for example if nodes (m, s) and $(m, -s)$ have the same parent in in the first node list, then it can be safely concluded that all other nodes (m, r) with $|r| < s$ have the same parent. Therefore the nodes in a level should be considered in the order $(m, t), (m, -t), (m, t-1), (m, -t+1), ..., (m, 0)$.

It is often the case that the parent node of a given current node is not uniquely defined. There may be several active nodes all of which provide a shortest path of the same length to the current node (m, s). We suggest to base the choice on minimizing the entropy for the encoding of the PL approximation as follows.

Most digital signals have samples taken at equal intervals, therefore let us assume from now on that $x_m = m$ for $m = 1, ..., N$. In this case coding a PL approximation $\tilde{f}(x) = \varphi(x; x_{i_1}, \tilde{y}_{i_1}, ..., x_{i_K}, \tilde{y}_{i_K})$ obviously is equivalent to encoding the integer vectors $(i_1, ..., i_K)$ and $(\tilde{y}_{i_1}, ..., \tilde{y}_{i_K})$. In place of these vectors it is advantageous to use differential coding and instead encode the difference vector $\mathbf{x} = (i_1, i_2 - i_1, i_3 - i_2, ..., i_K - i_{K-1})$ and $\mathbf{y} = (\tilde{y}_{i_1}, \tilde{y}_{i_2} - \tilde{y}_{i_1}, ..., \tilde{y}_{i_K} - \tilde{y}_{i_{K-1}})$. Since the components of the vectors $\mathbf{x}$ and $\mathbf{y}$ will have differing distributions we use entropy coding with two corresponding contexts, called x and y.

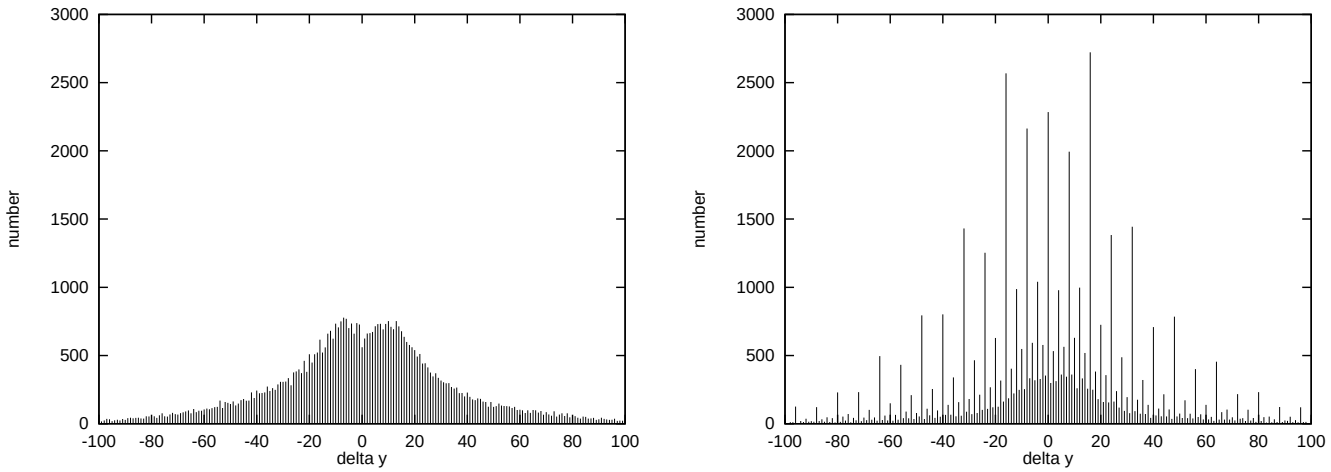


Figure 6. Histograms of frequencies for differential coding of intensity values. This graph is for the 512×512 test image Lenna and the L_∞ error bound is $t = 5$. The optimal PL approximation requires 50869 segments. In the right graph the strategy for choosing an optimal parent node according to maximum number of trailing zeros in the binary representation of the difference in y is employed. This way the entropy of 7.17 (left distribution) is reduced to 6.51 (right distribution).

Now, assume that for current node (m, s) there are several active nodes (i, r) which yield shortest paths to (m, s). In order to achieve a low entropy for the distribution of the intensity differences $\Delta\tilde{y}$ we propose to choose a node (i, r) such that the binary representation of the difference $\Delta\tilde{y} = m - i + s - r$ has a maximum number of trailing zeros. This technique reduces the entropy considerably. Figure 6 shows a comparison of distributions derived from this strategy and the default procedure, where the first optimal parent node encountered in the search is chosen.

5. MINIMUM ENTROPY PL APPROXIMATION BY ANNEALING

While the dynamic programming in the sections above yields PL approximations for a given error bound with the optimum number of line segments it cannot be concluded that this solution also provides a PL approximation with an optimal bit rate when entropy coding is used. In fact, in none of the cases we tested using our implementation the optimum PL approximation was also optimal with respect to rate. In this section we introduce an annealing technique which allows to find PL approximations with locally optimal rates. The problem addressed and the approach taken here is novel for piecewise linear coding, but in spirit similar to many iteration algorithms, an example of which is in the work of Ke and Marcellin,[2] where minimum entropy DPCM codes were considered. In this paper we describe the technique only briefly. A more detailed version with comprehensive simulation results will appear elsewhere.

Recall that the differential coding of a PL approximation $\tilde{f}(x) = \varphi(x; x_{i_1}, \tilde{y}_{i_1}, ..., x_{i_K}, \tilde{y}_{i_K})$ means encoding the integer vectors $\mathbf{x} = (i_1, i_2 - i_1, i_3 - i_2, ..., i_K - i_{K-1})$ and $\mathbf{y} = (\tilde{y}_{i_1}, \tilde{y}_{i_2} - \tilde{y}_{i_1}, ..., \tilde{y}_{i_K} - \tilde{y}_{i_{K-1}})$. Assume that we are given two variable length codes H_x and H_y for the components of the vectors $\mathbf{x}$ and $\mathbf{y}$. Then the codeword lengths for the differences $i_{k+1} - i_k$ and $\tilde{y}_{i_{k+1}} - \tilde{y}_{i_k}$ are integers, denoted by $\mathrm{len}(H_x(i_{k+1} - i_k))$ and $\mathrm{len}(H_y(\tilde{y}_{i_{k+1}} - \tilde{y}_{i_k}))$. We can now pose

Problem 2. *Let $P_i = (x_i, y_i)$, $i = 1, ..., N$ with $x_i = i$ for $i = 1, ..., N$ be N points in Z^2, where Z is the set of integers. Moreover, let $t \in \{0, 1, 2, ...\}$ be a given error bound. Let also two variable length codes H_x and H_y be given as above. Then the* Problem of rate-optimal PL approximation *is to find a PL approximation $\tilde{f}(x) = \varphi(x; x_{i_1}, \tilde{y}_{i_1}, ..., x_{i_K}, \tilde{y}_{i_K})$ of the data with error bound t such that the rate of the code for $\mathbf{x} = (i_1, i_2 - i_1, i_3 - i_2, ..., i_K - i_{K-1})$ and $\mathbf{y} = (\tilde{y}_{i_1}, \tilde{y}_{i_2} - \tilde{y}_{i_1}, ..., \tilde{y}_{i_K} - \tilde{y}_{i_{K-1}})$, i.e.,*

$$\mathrm{len}(H_y(\tilde{y}_{i_1})) + \sum_{k=1}^{K-1} \mathrm{len}(H_x(i_{k+1} - i_k)) + \mathrm{len}(H_y(\tilde{y}_{i_{k+1}} - \tilde{y}_{i_k}))$$

is minimal.

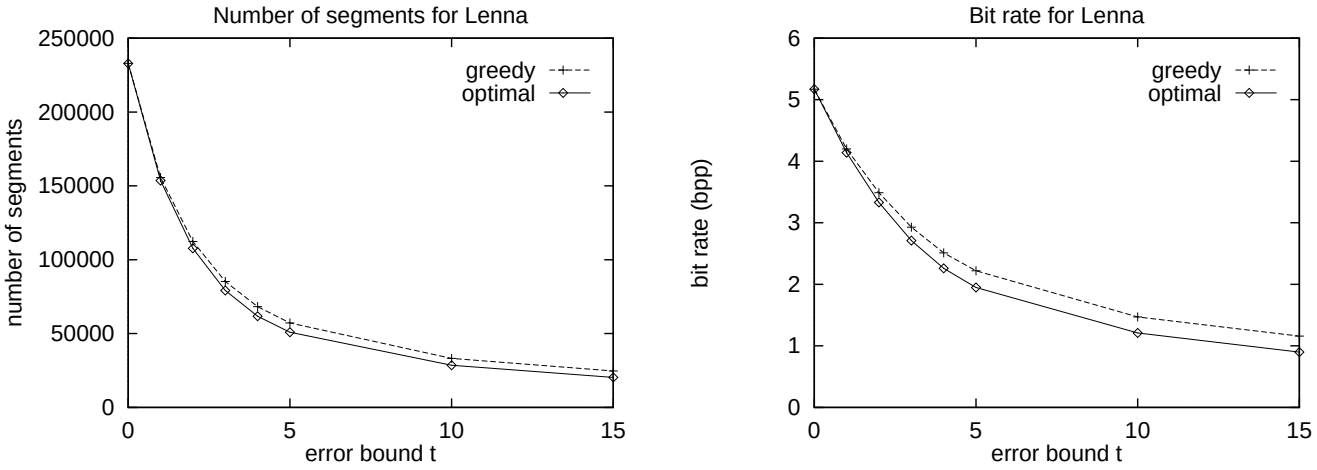


Figure 7. Number of intervals and bit rate as a function of L_∞ constraint t.

This problem can be solved using dynamic programming just like Problem 1 in the previous section. The procedure is identical, with the only change in equation (4), which now reads

$$\begin{aligned} F_t(m,s) &= \min_{(i,r)\in I_{m,s}} \{F_t(i,r) + \mathrm{len}(H_x(m-i)) + \mathrm{len}(H_y(y_m - y_i + s - r))\}, \\ F_t(1,s) &= \mathrm{len}(H_y(y_1 + s)). \end{aligned} \tag{6}$$

The rate optimal PL approximation assumes a priori given variable length codes H_x and H_y, which, however, may not necessarily be the optimal (Huffman) codes for the actual distribution obtained from the PL approximation. In this case, adapting the variable length codes to the obtained solution will improve the rate for the encoding. Thus, the next problem is to jointly optimize the PL approximation together with the variable length codes.

With the above it is natural to suggest an iterative algorithm for improving the rate for PL coding. Beginning with fixed length codes H_x and H_y we use dynamic programming to arrive at a rate-optimal PL approximation. This in turn defines new (Huffman) variable length codes for which the rate will typically be smaller. Now we can loop back and find the best PL approximation using the new codeword lengths and so forth. Note that the first step, using fixed length codes, is identical to the solution of Problem 1, minimizing the number of line segments.†

This algorithm is guaranteed to converge, since in each step the rate is non-increasing. However, there is no guarantee that a *globally* minimal rate can be achieved. After convergence we have that the rate can be reduced neither by changing the underlying variable length code nor by considering another PL approximation (but perhaps by changing both together). Such an entropy minimum can be called a local minimum.

6. RESULTS

We have applied the methods to several test images (shown in Figure 2) using both the line-by-line zig-zag image scan as well as the Hilbert scan. The zig-zag line scan processes lines sequentially and alternatingly in left-to-right and right-to-left fashion. The greedy search strategy[5,6] is compared to the optimal one as computed by our dynamic programming approach. We also report results of simulations of the annealing procedure for minimum length encodings. Our implementation is in the C++ language and the tests were performed using a 100 MHz Intel 486 processor.

6.1. Line scan versus Hilbert scan

We begin with a comparison of the two image scanning procedures: Zig-zag line scan versus the Hilbert scan. As the Hilbert scan results in a larger autocorrelation function of the one-dimensional signal (see Figure 4) we expect to see

† In an implementation care must be taken to handle the zero frequency problem well.

Results for Test Image Lenna										
	Greedy Algorithm				Optimal Algorithm				# of	
t	# of segments	rate (bpp)	rms error	time (secs)	# of segments	rate (bpp)	rms error	time (secs)	segments gained	in %
0	232944	5.17	0.00	3	232846	5.17	0.00	6	98	0.0
1	155634	4.20	0.83	4	153639	4.14	0.82	16	1995	1.3
2	112356	3.49	1.53	4	107704	3.33	1.51	32	4652	4.1
3	85295	2.93	2.19	4	79124	2.71	2.13	59	6171	7.2
4	68208	2.51	2.78	4	61703	2.26	2.68	100	6505	9.5
5	57176	2.22	3.34	4	50869	1.95	3.20	155	6307	11.0
10	33188	1.47	6.12	5	28533	1.21	5.83	559	4655	14.0
15	24724	1.16	9.07	6	20361	0.90	8.65	1245	4363	17.6

Results for Test Image Landsat										
	Greedy Algorithm				Optimal Algorithm				# of	
t	# of segments	rate (bpp)	rms error	time (secs)	# of segments	rate (bpp)	rms error	time (secs)	segments gained	in %
0	234039	4.65	0.00	4	234039	4.65	0.00	7	0	0.0
1	221399	4.95	0.74	5	221272	4.94	0.74	14	127	0.1
2	191777	4.81	1.39	6	191514	4.67	1.37	21	263	0.1
3	166181	4.51	2.30	6	164908	4.35	2.29	29	1273	0.8
4	152910	4.31	2.87	8	151375	4.12	2.83	38	1535	1.0
5	138779	4.11	3.52	8	136899	3.82	3.46	49	1880	1.4
10	84217	3.07	7.04	9	78490	2.73	6.86	150	5727	6.8
15	58117	2.32	9.96	10	51316	1.92	9.78	359	6801	11.7

Results for Test Image Angio										
	Greedy Algorithm				Optimal Algorithm				# of	
t	# of segments	rate (bpp)	rms error	time (secs)	# of segments	rate (bpp)	rms error	time (secs)	segments gained	in %
0	197590	4.12	0.00	3	197329	4.12	0.00	8	261	0.1
1	96306	2.87	0.84	3	92833	2.80	0.84	22	3473	3.6
2	68229	2.27	1.51	3	62779	2.14	1.48	41	5450	8.0
3	55846	1.97	2.17	3	49180	1.79	2.11	68	6666	11.9
4	47225	1.75	2.83	3	39546	1.53	2.72	106	7679	16.3
5	40170	1.56	3.46	3	31883	1.30	3.28	158	8287	20.6
10	19344	0.89	6.17	3	12335	0.61	5.56	910	7009	36.2
15	12420	0.63	8.66	3	7827	0.41	7.95	5486	4593	37.0

Table 1. Results for the greedy and optimal algorithm for test images. The total number of points in each image scan is $512^2 = 262144$. The rates in bpp are obtained using separate Huffman codes for the differential coding of the segment lengths and the intensity values. Rates in this table do not include overhead of the Huffman tables (about 600-1000 bytes per image).

that with the Hilbert scan the minimal number of segments in a PL approximation is smaller than for the line scan. However, this is not the case, see the table in Figure 4. Clearly, the minimal number of line segments is larger for the Hilbert scan and L_∞ error bounds up to $t = 4$. Only for larger tolerances of $t \geq 5$ does the stronger autocorrelation for the Hilbert scan provide an advantage for piecewise linear coding. Similar results were obtained for the other two test images Landsat and Angio (not shown here), also when replacing the optimal algorithm with its greedy version. Figure 5 provides a visual explanation showing that it is more likely to encounter linearly changing intensity values

	Iterated Greedy Algorithm				Iterated Optimal Algorithm					
	# of	bit rate	rms	time	# of	bit rate	rms	time	bit rate difference	
iterations	segments	bpp	error	secs	segments	bpp	error	secs	bpp	in %
0	85295	2.98	2.19	2	79124	2.73	2.13	61	0.25	8.4
1	98238	2.88	2.08	2	81553	2.51	2.02	93	0.37	12.8
2	102842	2.74	2.01	1	85748	2.48	1.98	93	0.26	9.5
3	105870	2.69	1.98	2	87079	2.47	1.97	94	0.22	8.2
4	106317	2.67	1.98	2	88104	2.46	1.97	94	0.21	7.5
5	106407	2.67	1.97	2	88331	2.46	1.97	95	0.21	7.5
gain	−21112	0.33	0.22		−9207	0.27	0.16			

Table 2. Results for the iterated greedy and optimal algorithms for Lenna and $t = 3$. These results are for the line scan. With the Hilbert scan the achievable bit rates are 0.10 bpp lower, both for the greedy and the optimal algorithm, yielding 2.57 and 2.36 bpp.

along part of a scan line rather than along the convoluted Hilbert curve. However, it is interesting to note, that in spite of the larger number of segments for the Hilbert scan entropy encoding of the PL approximations can be expected to yield a better overall bit rate. The same conclusion holds for the encodings of the Landsat image, but not for Angio, where the Hilbert scan requires many more segments than the line scan. Our results are similar to those found by Rosenberg[10] who compared many different scanning algorithms for greedy PL approximation schemes.

6.2. Optimum uniform PL approximations versus greedy algorithm

Since in this section the emphasis is on a minimal number of segments in PL approximations , we show only the results obtained with the line scan. Table 1 gives the results and Figure 7 displays the number of segments and the achieved bit rates for one of the images and both algorithms, greedy and optimal. The number of segments turns out to be smaller with the optimal method in all cases, as expected. In some cases the difference exceeds one thousand segments. Moreover, the table shows that the greedy search in fact is suboptimal by a significant amount also in terms of subsequent 0-order entropy encoding. The advantage of the optimal algorithm grows monotonically as the error bound t gets larger. The price to pay for the superior performance is the larger processing time. While the cpu time necessitated by the greedy algorithm is small and nearly independent of the choice of the error bound t, the optimal algorithm requires a greater amount of time, which, moreover, rapidly grows, as t increases.

6.3. Iterative improvement of bit rate by annealing

In this subsection we provide a part of our results using the minimum entropy annealing technique described in Section 5. Table 2 shows number of segments, bit rates, and so on for the test image Lenna and the error bound $t = 3$. Figure 8 corresponds to the table and displays the decreasing bit rates as the iterations proceed. Convergence up to 0.01 bits per pixel required 5 iterations. In order to achieve a bit rate reduction the number of segments grows from the initial minimal number of about 79000 to over 88000 using the optimal algorithm. The bit rate drops by 0.27 bpp which is about 10% of the rate achieved just by minimizing the number of segments. The root-mean-square error also improves during the annealing iterations.

Since this result is not globally optimal, the greedy algorithm together with the annealing iterations can potentially outperform the dynamic programming algorithm. In fact, the iterations provide an improvement of 0.33 bpp for the greedy algorithm, which is larger than the 0.27 bpp improvement that the annealing provides for the optimal algorithm. However, this is by far not enough the come close to the end result of 2.46 bpp.

6.4. A critical remark

Even though we do not intend to compare results with those obtained with other L_∞-distortion bounded image coding methods we wish to critically remark that the approach using an image scan with subsequent piecewise linear coding of the resulting 1-D signal cannot compete with state-of-the-art near-lossless image coders. For example, a nearlossless version of the CALIC coder[14] yielded a rate of only 1.6 bpp for the Lenna test image for the L_∞ error bound $t = 3$, while we obtained 2.36 bpp as our best result (using the Hilbert scan, dynamic programming, and

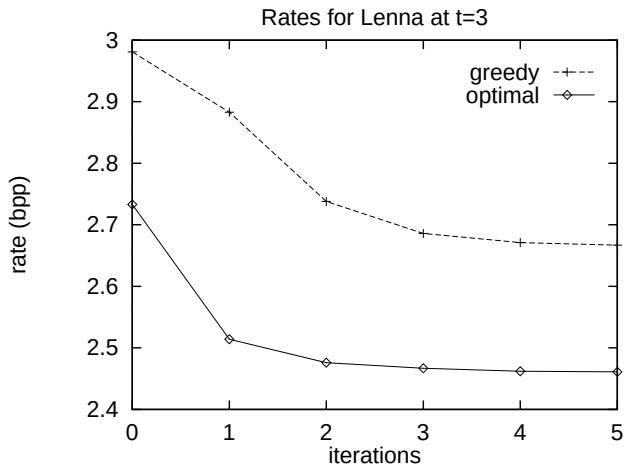


Figure 8. Bitrate as a function of the number of iterations, both for the greedy and the optimal algorithm.

annealing iterations). It seems unlikely, that the PL approach can be improved to match the 1.6 bpp performance of CALIC, even if sophisticated prediction (in place of the simple differential coding employed here) and context modeling (not used here) are introduced. The PL coding approach seems more natural for signals that are one-dimensional to begin with.

7. CONCLUSION AND FUTURE WORK

We have presented an optimal L_∞-distortion limited image encoding method based on image scans and piecewise linear approximations of the resulting one-dimensional signal. Our results improve those from previous methods, which employ a greedy strategy or constrain the break points more than necessary. Furthermore, we have extended the approach such that not the number of runs but the length of the corresponding entropy code for the runs and intensities is minimized. This can be achieved with an iterative annealing technique.

There are several open problems for future work:

1. The encoding of the PL approximation considered in this paper is simple. More advanced coding method such as context coding should be incorporated to lower the rate.
2. The greedy method can be extended by prescribing a larger depth for the search, for example, by looking ahead in the error tunnel using two or more line segments. Will this approach together with the iterative annealing be able to perform as well or better than the dynamic programming technique in less processing time?
3. Instead of piecewise linear approximation one may generalize to polynomial approximations.
4. The dynamic programming approach will become infeasible, when the resolution of the intensity values is increased. For example, for data with 16 bit intensity values tolerances of size $t = 300$ or even $t = 1000$ will be relevant. Clearly, this will lead to vast storage requirements and run times. Moreover, the encoding result cannot be expected to be good, because the PL approximation has to be specified with a resolution that is not required. In this case a quantization of the break points to a grid with a grid spacing larger than 1 may provide an appropriate solution for PL encodings. The tradeoff between the choice of the quantization step size, the encoding performance (rate), and processing time should be analyzed.

ACKNOWLEDGEMENTS

The author wishes to thank Maximilian Thiel for developing the computer code and running the simulations, Hannes Hartenstein for discussions on L_∞-distortion limited compression. Thanks also to Konstantin Konstantinides and Balas Natarajan for providing materials and insight in their original DCC'93 work. A part of this work was carried out while the author was with the Department of Computer Engineering at the University of California, Santa Cruz. This support is gratefully acknowledged.

REFERENCES

1. M. J. Gormish and J. T. Gill, "Discrete minimum entropy quantization," *IEEE Trans. Image Processing* **4,9**, pp. 1314–1317, 1995.
2. L. Ke and M. W. Marcellin, "Near-lossless image compression: minimum entropy, constrained-error DPCM," *Proc. IEEE ICIP'95* , 1995.
3. P. J. Hahn and V. J. Mathews, "Distortion limited vector quantization," in *Proceedings Data Compression Conference*, J. A. Storer and M. Cohn, eds., 1996.
4. T. Bially, "Space fillinh curves: Their generation and their application to bandwidth reduction," *IEEE Trans. Inform. Theory* **IT-15**, pp. 658–664, 1969.
5. F. Pinciroli, C. Combi, G. Pozzi, L. Portoni, M. Negretto, and G. Invernizzi, "Some experiments in compressing angiocardiographics images according to the Peano-Hilbert scan path," *Computer methods and programs in biomedicine* **43**, pp. 247–253, 1994.
6. V. Bhaskaran, B. K. Natarjan, and K. Konstantinides, "Optimal piecewise-linear compression of images," in *Proceedings Data Compression Conference*, J. A. Storer and M. Cohn, eds., 1993.
7. R. A. Fowell and D. D. McNeil, "Faster plots by fan data-compression," *IEEE Computer Graphics and Applications* **9,2**, pp. 58–66, 1989.
8. H. Hartenstein, R. Herz, and D. Saupe, "A comparative study of L_∞ distortion limited image compression algorithms," in *Proceedings Picture Coding Symposium, Berlin*, 1997.
9. J. G. Dunham, "Optimum uniform piecewise linear approximation of curves," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **PAMI-8,1**, pp. 67–75, 1986.
10. C. J. Rosenberg, "A lossy image compression algorithm based on nonuniform sampling and interpolation of the image intensity surface," MS Thesis, Department of Computer Science, Massachusetts Institute of Technology, 1990.
11. R. Bellman, "On the approximation of curves by line segments using dynamic programming," *Communications of the ACM* **4**, p. 284, 1961.
12. B. Gluss, "A line-segment curve fitting algorithm related to optimal encoding of information," *Information and Control* **5**, pp. 261–267, 1962.
13. H. Imai and M. Iri, "An optimal algorithm for approximating a piecewise linear function," *Journal of Information Processing* **9,3**, pp. 159–162, 1986.
14. X. Wu, W. K. Choi, and P. Bao, "L-constrained high-fidelity image compression via adaptive context modeling," in *Proceedings Data Compression Conference*, J. A. Storer and M. Cohn, eds., 1997.