RC22306 (W0201-077) January 15, 2002
Computer Science

IBM Research Report

A joint simplification and compression method for vector-based image representations

Laurent L. Balmelli
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

A JOINT SIMPLIFICATION AND COMPRESSION METHOD FOR VECTOR-BASED IMAGE REPRESENTATIONS

Laurent Balmelli

Visual and Geometric Computing
IBM Research, T.J. Watson Center

ABSTRACT

In this paper, we propose a new compression algorithm for vector-based image representations as used in Postscript or Macromedia Flash. We present a joint optimization method combining geometric simplification techniques and quantization in a single framework. Both optimization techniques are usually used independently. We propose an optimal algorithm in the operational rate-distortion sense to solve the problem. We show that our method gives up to 10 times higher compression ratios for the same quality compared to standard compression methods such as trellis coding.

1. INTRODUCTION

1.1. Motivation

Vector-based representations describe images using geometric primitives, such as polygons and splines (Figures 1a-b). For example, a polygon is used to render the wing of the fly shown in Figure 1a. Compared to their bitmap equivalents, vector-based representations are more compact and can be, for example, scaled without loss of resolution. They are used by several graphic languages such as Postscript and Macromedia Flash[1]. The latter takes advantage of their compactness to deliver animations over the Internet (Figure 1a). *Contour maps*, as used in Cartograpy, can also be seen as vector-based representations. In this context, maps are formed by large sets of curves connecting terrain point at constant elevation.

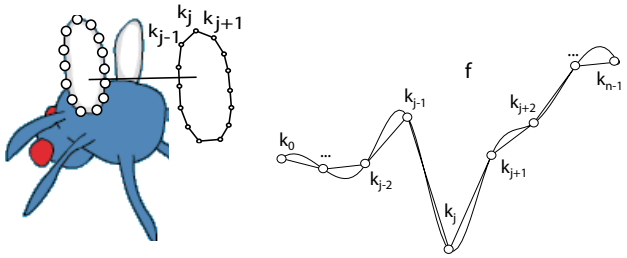

Fig. 1. (a) Image rendering using a vector-based representation. (b) Interpolated set of knots.

A common way to compress a vector-based representation is to quantize the coordinates of its primitives, for example using *trellis coding* [1]. Hence, each knot k_j forming the polygon in Figure 1a is quantized and the fly model is compressed by considering each primitive independently.

On the other hand, several *simplification methods* have been developed in Cartography and Computer Graphics [2, 3]. For example, these techniques are used to reduce the number of elevations for describing contour maps. Typically, contour maps are large datasets and their efficient processing requires to lower the amount of data. Given a polygon of n knots, a simplification algorithm computes an approximation by discarding knots [3]. The resulting polygon should represent the original one as faithfully as possible. The optimal solution with respect to the l_2 norm is given using dynamic programming. However, computationally efficient greedy approaches can be used in order to process large datasets. We refer the interested reader to [4] for a comparison between optimal and greedy simplification techniques.

Quantization and simplification, as described above, are complementary techniques and are usually applied independently. For example, a vector-based representation can be simplified, then quantized as a second step. Compression techniques, such as trellis coding, do not exploit the geometric redundancy of encoded primitives. For example, these methods do not reduce the number of knots in oversampled regions. On the other hand, simplification techniques do not allow the user to control the final bitrate of the representation since only a target number of knots can be specified.

1.2. Contributions and plan

In this work, we propose to perform *jointly* simplification and quantization. Our algorithm jointly selects a set of knots and a set of quantizers for these knots in order to encode primitives for a target number of bits. To the author's knowledge, such joint optimization framework has not been previously proposed. We study the optimal solution to the problem, i.e. such that the compressed representation has the lowest distortion for a given bitrate. In other words, our approximations are optimal in the operational rate-distortion sense. Our experimental results show that our method gives up to 10 times higher compression ratios for the same quality compared to standard compression methods such as trellis coding [1].

The paper is organized as follows: In Section 2.2 we present an optimal simplification algorithm. Then in Section 2.3, we explain how to extend this algorithm in order to solve the joint optimization problem. We analyze the complexity of our algorithm and discuss extensions in Section 2.4. We give extensive experimental results in Section 3 and conclude this paper in Section 4.

[1]Postscript is a trademark from Adobe. Flash is a trademark from Macromedia.

2. JOINT APPROACH TO SIMPLIFICATION AND COMPRESSION

2.1. Framework

Our algorithm applies to any type of primitive constructed on a set of knots $k_j \in \mathbb{R}^2$. These knots are interpolated in order to be rendered (Figures 1a-b). Languages such as Postscript or Flash use splines of varying order as interpolants. Hence, knots can form closed shapes as well as individual curves. Figure 1b shows a set of knots interpolated both with linear and cubic splines. We denote by f the resulting curve.

Consider a set of knots $\{k_j\}$ with $j = 0 \dots n-1$ and call f the curve interpolating the set (Figure 1b). Consider further a set of quantizers $\{q_k\}$ with $k = 0 \dots m-1$, each working at integer bitrate b_k. We call $q_k(k_j)$ the knot resulting from the quantization of k_j with q_k.

Let $\mathbf{j}$ be a sequence of l indices in $0 \dots n-1$ and $\mathbf{i}$ be a sequence of l indices in $0 \dots m-1$. Then, $\hat{f} = f(\mathbf{j}, \mathbf{i})$ is the approximation constructed on knots $k_{j \in \mathbf{j}}$, where each knot is associated with a quantizer $q_{i \in \mathbf{i}}$. We denote by $D(\hat{f})$ the distortion of $\hat{f}$, whereas $R(\hat{f})$ measures the total bitrate. The distortion $D(\hat{f})$ is computed as the distance in l_2 norm between the input curve f and an approximation $\hat{f}$. The rate $R(\hat{f})$ is obtained by summing the number of bits used to quantize each knot forming $\hat{f}$.

Given a budget of C bits, the joint optimization problem to solve is

$$\begin{aligned} &\min_{\mathbf{j},\mathbf{i}} D(\hat{f}), \\ &R(\hat{f}) = C. \end{aligned} \tag{1}$$

In other words, we want to select of set of l knots, given by indices in $\mathbf{j}$, and assign a quantizer to each knot. The assignment is given by the quantizer indices in $\mathbf{i}$. The approximation $\hat{f}$ constructed on the l quantized knots satisfies the bit budget C at minimal distortion. To solve the problem, we proceed in two steps: First, we present an optimal algorithm to select a set of knots in f in order to generate an approximation $\hat{f}$ (Section 2.2). The algorithm, based on dynamic programming, is optimal in the sense that $D(\hat{f})$ is minimized for a target number of knots l. Second, we explain how to include the quantizer assignments in our framework, i.e. we give an algorithm to solve (1) (Section 2.3).

2.2. Simplification algorithm

Our simplification algorithm is based on dynamic programming and uses a causal approach to select the knots. In our example, we consider the input primitive to be an open curve formed by a set of knots k_j linked by a linear interpolant (Figure 1b). The same algorithm can be applied to closed shapes. We discuss the choice for the interpolant in Section 2.4. The algorithm returns all optimal approximations $\hat{f}$ of f using 2 to n knots.

We use a trellis approach to model the knot selection (Figures 2a and 3). Each step of the algorithm is represented by a column in the trellis. The nodes in each column represent the knot indicated at the top. A link between two nodes corresponds to a segment (pair of interpolated knots) in $\hat{f}$. Hence, a path in the trellis (i.e. series of links) represents an approximation $\hat{f}$. Note that only paths linking knots k_0 and k_{n-1} cover the domain of f. We call *partial* an approximation not connecting these nodes. Finally, each row counts the number of knots in a path connecting a node in the row.

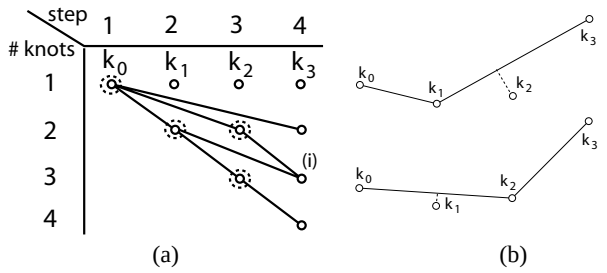

Fig. 2. (a) Trellis at step 4: The circles nodes represent the paths to update. (b) Approximations corresponding to the paths entering node (i).

The algorithm works as follows: At each step, knots k_j with increasing j are examined. Consider the trellis is Figure 2: At step 1, only the configuration $\{k_0\}$ is formed. Then, at step 2, the link between k_0 and k_1 is the partial approximation $\{k_0, k_1\}$. At step 3, we have the approximations $\{k_0, k_2\}$, $\{k_0, k_1, k_2\}$. Following the above examples, the new paths for each subsequent steps are formed as follows: Each previous paths (represented by nodes with incident links at preceding steps, see the circled nodes in Figure 2a) are connected to a node in the current step column. The node to connect is always one row below the one ending the path to be updated, i.e. each step increases by one the number of knots in each path. Hence at step 4, the approximations $\{k_0, k_3\}$, $\{k_0, k_1, k_3\}$, $\{k_0, k_2, k_3\}$ and $\{k_0, k_1, k_2, k_3\}$ are created (Figures 2a.)

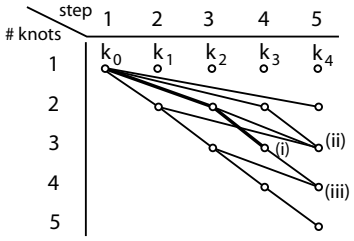

Fig. 3. Trellis at step 5: The bold lines correspond to the retained path at step 3 (Figure 2a.)

In order to avoid an exponential growth of knots configurations, the algorithm compares at each step paths having the *same number of nodes entering the same node*. In other words, it compares partial approximations having the same number of knots, covering the same domain. At each node, only the path incurring minimal distortion is retained. For example, the node (i) in Figure 2a is reached by the approximations $\{k_0, k_1, k_3\}$ and $\{k_0, k_2, k_3\}$. The corresponding partial approximations are shown in Figure 2b. In the figure, the errors incurred when discarding knots k_1 and k_2 are suggested using dotted lines. Following the above example, the approximation $\{k_0, k_2, k_3\}$ incurs minimal distortion. Hence, only the corresponding path is retained. At step 5 (Figure 3), all

remaining paths from previous steps are now connected to a node in k_4's column. The retained path from step 4 is represented using bold lines in the figure. Note that at this step, three paths of length 3 and two paths of length 4 must be compared (see nodes (ii) and (iii) in Figure 3, respectively.)

The algorithm is iterated until all paths connect knot k_{n-1}, i.e. the last knot in the curve (Figure 1b). We briefly review the complexity of the algorithm in Section 2.4. It is shown in [4] that the algorithm still explores all possible solutions and returns the optimal approximations. Starting from any node in the last column of the trellis (corresponding to the last step), an approximation $\hat{f}$ is found by backtracking its path until k_0 is reached. The number of knots in $\hat{f}$ depends on the row from which the path is started. We can choose to reconstruct any approximation having from 2 (i.e. using one single segment) to n knots.

2.3. Joint approach

We explain now how quantization can be included in our framework in order to perform joint simplification and compression. We show how the trellis structure presented in Section 2.2 can be modified in order to find the solution to (1).

Assume that a set of m quantizers $\{q_0, \ldots, q_{m-1}\}$ is available. Recall the causal approach used for selecting the knots, as explained in Section 2.2. Then at each step, we can choose between m quantized versions of the knot to construct $\hat{f}$. Hence, the number of nodes in each column of the trellis is multiplied by the number of quantizers. In Figures 4a-b, we give an example with two quantizers q_0 and q_1. The columns now contain twice the nodes as they had in the trellises of Figure 2a and 3. Also, each row now counts the number of bits in the approximation. In order for each path entering a node to have the same rate, we need to impose a constraint on the bitrates b_k of the quantizers q_k. Let $b_0 = b$ denote the minimal rate, then all quantizer rates must be multiples of b, i.e. $b_k = (k-1)b$, with $k = 0 \ldots m-1$. Hence, each row corresponds to approximations having bitrates multiples of b (Figures 4a-b).

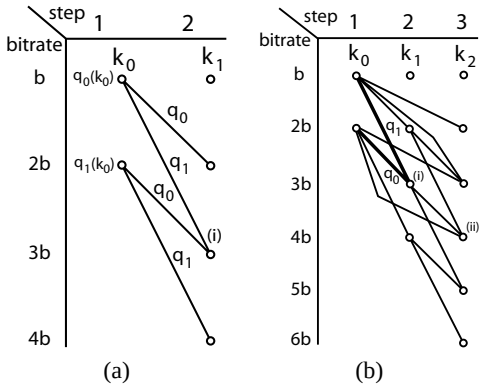

Fig. 4. Joint simplification and quantization: Trellis after (a) step 2 and (b) step 3. The bold lines at step 3 represent two remaining approximations from step 2.

We explain now the modified algorithm: In the new trellis, columns still represent knots k_j. However, the quantizer used for a knot is determined by *the link entering its node* (Figure 4a). The only exception is k_0's column, since no links enter nodes in this column. At step 1, configurations $\{q_0(k_0)\}$ and $\{q_1(k_0)\}$ are formed. Then, at step 2 (Figure 4a), we have now four approximations: $\{q_0(k_0), q_0(k_1)\}$, $\{q_0(k_0), q_1(k_1)\}$, $\{q_1(k_0), q_0(k_1)\}$ and $\{q_1(k_0), q_1(k_1)\}$. The new algorithm compares paths having the same rate entering the same node. However, *only paths using the same quantizer at the ending node can be compared*. This additional condition is important, as explained in the following example: In Figure 4a at node (i), we have two approximations of rate $3b$, i.e. $\{q_0(k_0), q_1(k_1)\}$ and $\{q_1(k_0), q_0(k_1)\}$. However, they cannot be compared because one ends at knots $q_1(k_1)$, whereas the other ends at knot $q_0(k_1)$. These knots have actually different values after quantization, which prevents comparison.

One more step of the algorithm clarifies how partial approximations are compared. In Figure 4b at the node (ii), we have the approximations:

(a) $\{q_0(k_0), q_1(k_1), q_0(k_2)\}$, (b) $\{q_0(k_0), q_0(k_1), q_1(k_2)\}$,
(c) $\{q_1(k_0), q_0(k_1), q_0(k_2)\}$, (d) $\{q_1(k_0), q_1(k_2)\}$.

Note that, even though only three links enter node (ii), the two links entering node (i) (bold lines in Figures 4b) induce approximations (a) and (c) (above). These two approximations can now be compared without ambiguity, since they have the same rate ($4b$), same number of knots, and end at the same quantized knot. The algorithm is again iterated until all paths connect a quantized version of the last knot k_{n-1}. An approximation of any rate is constructed by backtracking the appropriate path. Each node stores the result of a comparison, which allows for reconstructing paths without ambiguity. For example, the results for the comparison at node (ii) in Figure 4b allows for selecting the correct outgoing link (bold lines) when backtracking through node (i).

2.4. Analysis

We give now an intuitive argument for the complexity of the simplification algorithm. A detailed analysis can be found in [4]. At each step, the number of nodes in the trellis' columns grows linearly. Moreover, all nodes in previous steps connect to exactly one node in the current step column. Hence, the cost at each step is quadratic and the price to compute all optimal approximations with any number of knots is cubic. In practice, if only the approximation satisfying a target number of nodes is needed, then the cost is quadratic. This analysis holds for our joint optimization algorithm: The number of nodes in the trellis' columns still grows linearly, however with a factor proportional to the number of quantizers. Hence, the joint optimization do not increase the complexity.

This cost is acceptable for primitives having a small number of knots (such as the ones used in Flash models). However, it might be too expensive for large datasets such as contour maps. In this case, greedy strategies such as [2] can be used to perform an initial segmentation of the primitives. Then, our algorithm can be applied to each segment in order to obtain piecewise-optimal approximations. In this case, the cost is linear with respect to the number of knots in the primitives.

In our solution, we assume that a linear interpolant is used to connect the knots. This provides a good approximation for the

simplification error and is sufficient for most applications. However, higher degree interpolants can easily be handled in our trellis structure. With higher degree spline interpolants, the comparison between partial approximations is delayed by a number of steps proportional to the support of the spline basis. Since spline bases have local support, the complexity of the algorithm remain unchanged.

3. EXPERIMENTAL RESULTS

Human-created vector models, such as the fly in Figure 1a, are usually composed of smooth primitives. On the other hand, datasets in Cartography are likely to contain noise due to the acquisition process. We show that our method is efficient in both cases. To do so, we use a Markovian model (2) to generate 100 sequences of 500 knots each. The model allows us to control the "amount" of high-frequencies in the representation. These sequences are interpolated using a linear spline for the experiments. Hence, we use the recurrence equation

$$\begin{aligned} k_0 &= 0, \\ k_i &= \rho k_{i-1} + \mathcal{N}(0,1), \end{aligned} \tag{2}$$

where $\mathcal{N}(0,1)$ is a normal random variable with zero mean and unit variance. The parameter ρ controls the correlation of the sequence $\{k_j\}$. Typically, values for ρ close to 1 return smooth sequences, i.e. similar to human-created shapes. On the other hand, values for ρ close to 0 return random sequences, i.e. noisy datasets. Note that the smoothness of sequences decreases very rapidly as ρ diminishes. Sequences constructed with $\rho = 0.9$ are already too noisy to be generated by a human.

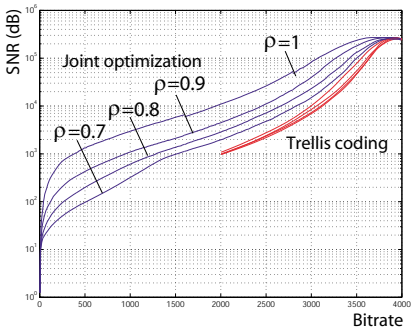

Fig. 5. Signal-to-noise ratio of the rate-distortion curves obtained with $\rho = 1, 0.9, 0.8, 0.7$. The bottom curves (half-sized) are obtained using standard trellis coding.

We use a set of two quantizers q_0 and q_1 having bitrates 4 and 8 bits, respectively. We compare our results to a standard trellis coding algorithm [1], i.e. no simplification is performed. Figure 5 shows the signal-to-noise ratio (SNR) of the rate-distortion (RD) curves obtained with $\rho = 1, 0.9, 0.8, 0.7$. Figure 6 shows the SNR of the RD curves obtained with $\rho = 0, 0.2, 0.4$. In both cases, the RD curves are the average of the results obtained with our 100 sequences. In Figures 5 and 6, the top curves are obtained with our algorithm, whereas the bottom curves are obtained using trellis coding. Note that we observe few differences in performance with the latter technique as ρ varies. For the curves obtained using trellis coding, the minimal and maximal rates are 2000 and 4000 bits, respectively, since we use sequences of 500 knots and two quantizers of 4 and 8 bits.

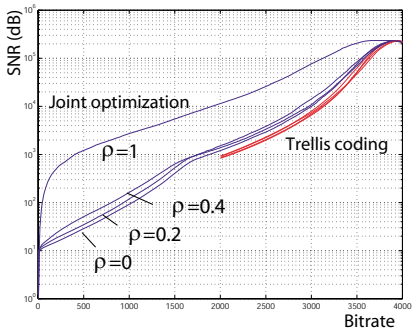

Fig. 6. Signal-to-noise ratio of the rate-distortion curves obtained with $\rho = 0, 0.2, 0.4, 1$. The bottom curves (half-sized) are obtained using standard trellis coding.

4. CONCLUSION

As shown in Figure 5, our method is efficient for smoothly varying curves. At high rate, the flat shape of the RD curve obtained with $\rho = 1$ shows that rate can be reduced by about 10% without loss in quality. The RD curves show more than 1 dB improvement compared to standard compression methods. In Figure 6 we can see that, for sequences containing mostly high-frequencies, our method shows less improvements. This result is expected however, since rapidly varying curves require more knots in order to be accurately represented. Another advantage of our method is that a wider range of possible target rates is available. Hence, higher compression ratios can be achieved. For example in Figure 5, we can see that the quality achieved by standard trellis coding at 2000 bits is obtained with our method at a rate of approximatively 200 bits for $\rho = 1$, i.e. we have a 10 times higher compression ratio.

In this work, we have shown that the performance of compression algorithms for vector-based representations can be improved by combining simplification and quantization in a joint framework. We give an efficient algorithm based on a trellis structure. Our method returns the optimal approximations in the operational rate-distortion sense.

5. REFERENCES

[1] C. Schlegel, *Trellis coding*, IEEE press, 445 Hoes Lane, P.O box 1331, Piscataway, NJ 08855-1331, 1997.

[2] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The Canadian Cartographer*, vol. 10, no. 2, pp. 112–122, 1973.

[3] L. Boxer, C. Chang, R. Miller, and A. Rau-Chaplin, "Polygonal approximation by boundary reduction," *Pattern Recognition Letters*, vol. 14, pp. 111–119, 1993.

[4] L. Balmelli, "Rate-distortion optimal mesh simplification for communications," *Phd dissertation No 2260*, vol. Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland., 2001.