Shape Coding using Temporal Correlation and Joint VLC Optimization

Gerry Melnikov, Guido M. Schuster* and Aggelos K. Katsaggelos

Northwestern University

Department of Electrical and Computer Engineering

McCormick School of Engineering and Applied Science

Evanston, Illinois 60208-3118

Tel: (847) 491-7164

Fax: (847) 491-4455

E-mail: {gerrym,aggk}@ece.nwu.edu

*3COM, Carrier Systems Business Unit

Advanced Technologies Research Center

1800 W. Central Rd.

Mount Prospect, Illinois 60056-2293

Tel: (847) 222-2486

Fax: (847) 222-0573

E-mail: guido_schuster@3com.com

Abstract

This paper investigates ways to explore the between frame correlation of shape information within the framework of an operationally rate-distortion (ORD) optimized coder. Contours are approximated both by connected second-order spline segments, each defined by three consecutive control points, and by segments of the motion-compensated reference contours. Consecutive control points are then encoded predictively using angle and run temporal contexts or by tracking the reference contour. We utilize a novel criterion for selecting global object motion vectors, which improves efficiency. The problem is formulated as Lagrangian minimization and solved using Dynamic Programming (DP). Furthermore, we employ an iterative technique to remove dependency on a particular VLC and jointly arrive at the ORD globally optimal solution and an optimized conditional parameter distribution.

I. Introduction

The object-oriented treatment of video data has regained its popularity with the advent of new multimedia applications. Such applications include content-based storage and retrieval, mobile communications, and film authoring. Within the object-oriented framework, a video sequence is represented through the evolution of video object planes (VOP), with each frame composed of one or more VOPs. Evolution of these VOPs in time is described in terms of shape, texture, and motion information. The optimal allocation of the available resources within these three components is a key fundamental problem addressed by the operational rate distortion theory (ORD). The shape component, particularly in very low bit rate applications, requires high efficiency of representation, since it takes up a significant percentage of the bit budget. In MPEG-4 the task of encoding the shape information is completely decoupled from those of motion estimation, texture coding, and boundary estimation. This is also the approach taken here, that is, we assume that the input to the proposed shape coder is a binary mask defining the objects in every frame.

In the process of evaluating competing techniques for the MPEG-4 standard, several binary shape coders were considered. These coders, however, lack optimality in their both intra and inter modes of operation. The context-based (CAE) coder [1] capitalizes on temporal redundancy by performing object-based motion compensation and extending the context template into the neighboring pixels of the reference frame. Similarly, the MMR inter-coder [10] differs from its intra mode counterpart in the choice of pixels serving as context. In the baseline and the vertex-based polynomial approaches (inter mode) [3], [5] a contour in the current frame is approximated through motion compensation by a contour in the previous frame, with segments exceeding a certain error threshold coded in their respective intra mode. All of these coders are ad-hoc in the intra mode, and, therefore, also in the inter mode. They fail to achieve operational optimality since they neither take the tradeoff between the rate and the distortion into account nor do they use the distortion metric used for their evaluation in the encoding process. In fact, whereas the four methods proposed in [1], [10], [3], and [5] achieve good compression in the intra mode, poor coding efficiency in the inter mode remains their major weakness.

We have previously proposed optimal approximations of a given boundary based on curves of different orders and for various distortion metrics, processing each frame independently (intra mode) [2]. In [4] this problem was solved optimally and jointly with the variable-length code selection. In this work we extend this ORD optimal framework to take into account the temporal contour redundancies present in typical video sequences. We employ a novel criterion for global object-based motion vector selection which fits naturally into the chosen code structure. We adaptively switch between context and tracking modes to better capitalize on temporal redundancies.

In addition to arriving at the inter mode ORD optimal representation of a sequence for a particular coding framework, characterized by fixed VLC tables, we employ an iterative procedure to find the underlying parameter probability distribution resulting in the locally most efficient ORD curve.

This paper is organized as follows. The algorithm structure is presented in Sec. II. The framework for taking advantage of frame to frame correlation between contours is addressed in Sec. III. Section IV-A deals with the context-based control point encoding scheme, while the additive distortion metric is discussed in Sec. IV-B. Section V describes how the problem can be formulated as a shortest path problem and Sec. VI discusses VLC optimization issues. Finally, results are presented and discussed in Sec. VII.

II. The Boundary Encoding Problem

In this paper we solve the problem of encoding temporally correlated contours optimally in the ORD sense within the chosen vertex-based framework. Contours are approximated by connected 2^{nd}-order B-spline segments, each defined by 3 consecutive control points, (p_{u-1}, p_u, p_{u+1}), with two neighboring splines sharing two control points. Thus an ordered set of control points constitutes a code for a (lossy) shape approximation. A 2^{nd}-order B-spline, henceforth referred to simply as a spline, is a parametric curve (parameterized by t) that starts at the midpoint, also called a knot, between p_{u-1} and p_u and ends at the midpoint between p_u and p_{u+1} as t sweeps from 0 to 1. Mathematically, it is defined as follows:

$$Q_u(p_{u-1}, p_u, p_{u+1}, t) = \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 & -1.0 & 0.5 \\ -1.0 & 1.0 & 0.0 \\ 0.5 & 0.5 & 0.0 \end{bmatrix} \cdot \begin{bmatrix} p_{u-1,x} & p_{u-1,y} \\ p_{u,x} & p_{u,y} \\ p_{u+1,x} & p_{u+1,y} \end{bmatrix}, \tag{1}$$

where $p_{i,x}$ and $p_{i,y}$ are the x and y coordinates of p_i, $i = u-1, u, u+1$. A sequence of 2^{nd}-order B-splines solves the interpolation problem at the knots, while being differentiable everywhere, including the knots. This smoothness property, coupled with the simplicity of definition, makes B-splines a natural choice for the shape coding application. Note that, in principle, we can force a spline to approximate a straight line by merging two neighboring control points.

The determination of the number and location of the control points is central to our approach. Although an ordered set of control points defining approximating splines may include any point in the image plane, it is unlikely that locations far from the original boundary would lead to an ORD optimal approximation. In addition, as will be seen in Sec. V, the complexity of the algorithm designed to find an ORD optimal solution to this problem is proportional to the square of the number of candidate control points. These two considerations lead naturally to the concept of the admissible control point band [8], [9], that is, the set of candidate control points. Figure 1A demonstrates a fixed width control point band drawn around the original boundary of a sample object in our test sequence. Note, however, that although the control points are restricted to belong to the admissible control point band, there is no restriction for pixels of the approximating contour to be inside the band. We label all pixels of the original contour based on their scanning order in which they were extracted from the binary mask specifying the object. Every admissible control point is then labeled with the label of the original contour pixel closest to it, as depicted in Fig. 1B. Imposing the requirement that consecutive control points are in increasing order ensures that we always move forward along the boundary.

III. Temporal Correlation

It is intuitively clear that object boundaries between frames are correlated. However, efforts to gain coding efficiency based on this apparent redundancy have, so far, met relatively small success [2]. In the proposed context methods [1], [10], one global motion vector, which minimizes the number of mismatched pixels, is employed to align corresponding objects in two consecutive frames. A context for a pixel in the current frame is then computed from its spatio-temporal neighborhood. Consequently, these algorithms in the inter mode differ from their intra mode counterparts only in the choice of neighboring pixels serving as contexts. The main disadvantage of this type of approach is its pixel-based nature, which suffers from mis-alignments due to motion and noise, as illustrated in Fig. 2. Clearly, the motion model used for this example performs poorly under non-rigid object motion. An additional important problem, however, is the contour degradation by noise, which makes predictions of whether a given pixel is on or off the boundary highly unreliable. Noise is introduced to contours during the frame acquisition and segmentation processes, which causes contours in consecutive frames to be different even without motion.

In the previously proposed intra mode [8], consecutive control point locations are decorrelated using a 2^{nd}-order prediction model, where every control point is encoded in terms of the relative angle α and the run length β, henceforth referred to as *run*, (in pixels), as depicted in Figs. 3A and 3B.

A. Control Point Encoding Using Context

In this work we reduce the effects of contour noise by utilizing temporal context for the predictive encoding of the (*angle*, *run*) symbols, instead of the underlying pixels. Thus, instead of using temporal context to predict boundary pixel locations, we use them to estimate the current (*angle*, *run*) symbol, which defines the location of the next consecutive control point. The underlying assumption is that these symbols are affected by noise to a much smaller extent than the original boundary. The (*angle*, *run*) framework for encoding consecutive control point locations was used in [4], [9] in conjunction with B-splines to arrive at an ORD optimal representation of a boundary in the intra mode. The context for the *angle* and *run* components is searched for in a local window in the motion compensated reference frame and are computed for every pixel in the admissible control point band, as a preprocessing step. Figure 4 depicts a hypothetical context window in the reference frame after motion compensation, which is centered on a pixel, denoted by O in the admissible control point band. It is used to extract both the most likely direction and the most likely length of the vector pointing from that pixel to the next potential control point. That is, if an actual control point is located at the current position, this context provides an estimate of where the next control point is most likely to be. The context for the *angle* component is obtained by selecting the direction in which most of the transitions between consecutive boundary pixels in the reference frame occur. This corresponds to the North-West (NW) direction in Fig. 4, which is pointed to 6 times. Then follow the North and South-East directions with 2 transitions each. Note that there is no ambiguity between diametrically opposite directions since all pixel transitions occur in the direction of the increasing label. This estimate of the direction, being a statistical average, is very robust to contour noise, as it tends to cancel noise, while determining the dominant direction. Similarly, the run length component of the context is obtained by selecting the most frequently occurring distance between consecutive control points in the context window. In this example, a run length of 4, occurring 3 times, is selected. For both the run and the angle, ties are broken in favor of the corresponding context with the smaller index.

Having computed contexts as described above, we employ a spatially adaptive VLC scheme that assigns shorter codewords to combined $(angle,\ run)$ symbols that are close to $(angle_{context},\ run_{context})$. Due to motion and occlusions, however, certain parts of the boundary will have too few reference pixels for a meaningful computation of the context, in which case the algorithm reverts to the intra mode ([4]) for encoding the $(angle,\ run)$ symbol, i.e., based on the previously encoded symbol.

With non-homogeneous and non-rigid motion it is often the case that certain contour segments are well approximated by the motion-compensated reference frame, while others are not. For this reason we include, in the source alphabet, symbols representing the tracking of pixels in the reference boundary. Thus, based on the chosen tradeoff between the rate and the distortion, the encoder may select to approximate stretches of the contour under consideration by following the reference contour for n pixels, with each value of n corresponding to one symbol. Since the previously reconstructed frame is available at the decoder, the next control point can be unambiguously determined by following the reference contour for a specified number of pixels. For example, we expect portions of the boundary in the feet area of the kid in Fig. (2) to be approximated through these symbols. These *tracking* symbols are discussed in more detail in Sec. IV-A.

B. Motion Estimation and Compensation

The issue of selecting a suitable global motion vector between two binary masks is a non-trivial one. Clearly, the optimal selection of the motion vector requiring the coupling of the motion estimator with the encoder, is not a practical approach. While motion models of varying complexity can be tried, the often overlooked question is that of the selection or matching criterion. All approaches evaluated by MPEG-4 [2] use a single global motion vector per object minimizing the number of pixels in error between the current and the reference objects. That is, a vector $\bar{M}$ is chosen such that

$$\bar{M} = arg \min ||E_{\bar{M}}||, \tag{2}$$

where $E_{\bar{M}}$ is the set of all pixels in error under motion compensation of the reference contour by $\bar{M}$. The above criterion often spreads the error pixels all around the boundary when object motion is non-rigid, which is inconsistent with the objective of tracking the reference contour or utilizing context where possible. To better capitalize on the proposed code structure (context and tracking) the following criterion is used to choose a global motion vector:

$$\bar{M} = arg \min \sum_{(i,j) \in E_{\bar{M}}} (i - \bar{i})^2 + (j - \bar{j})^2, \tag{3}$$

where the summation is over all pixels in error whose center of mass is at $(\bar{i}, \bar{j})$. Roughly speaking, application of the proposed criterion has the effect of pushing the majority of error pixels to one side of the contour, while accurately approximating the rest. This, together with the tracking mode and the use of context, makes the encoder efficient with respect to non-rigid object motion and contour noise. Figure 5 illustrates the application of the proposed criterion for motion. Here the contour in the current frame is shown by the solid line and the motion compensated (under the proposed criterion) contour is shown by the dotted line. Also, the resulting control points are shown by $*$ and the control points where the current boundary is approximated by tracking the reference boundary are shown by $\circ$.

IV. Rate and Distortion

So far we have concentrated on how to capitalize on temporal correlation between frames. Now that we have identified the code structure, we need to define the rate and the distortion associated with various code symbols.

A. Rate

Once the *angle* and *run* length contexts are known, the location of the next control point of the object in the current frame is encoded by (*angle*, *run*) predictively with respect to the context. In this framework, shorter codewords are assigned to directions and runs closest to the contexts. Figure 6A shows a typical conditional direction probability distribution, given the NW context, which also served as the initial distribution used in the VLC optimization procedure, discussed in Sec. VI. A total of 16 different directions are allowed, with the direction corresponding to the direction of the context having the greatest probability. Note that this distribution is conditional and distributions for the other contexts of direction are obtained by rotating the figure accordingly. Note also that with the proposed structure of the *angle* component, twice as many directional transitions are now possible than in [2], since we added the angles corresponding to a knight-like (as in chess) move of k pixels along one of the axis and a move of $2k$ along the other. In view of the VLC optimization (Sec. VI), this is not, however, going to have an adverse impact on the rate,

Figure 6B shows the 8 equiprobable directions when a location under consideration does not have an angle context. In this case, the encoder reverts to the intra mode style, i.e., the current direction is encoded predictively from the previously encoded one. Overall, the angle alphabet has 16 and 8 symbols for the two situations, respectively. Similarly, an initial guess for the conditional probability distribution for the run length is shown in Fig. 7A for the case the context is 4. Since there are 5 different run lengths for 5 possible contexts, a total of 25 *run* symbols are used when the *run* context is present, and another 5 symbols are used when it is not. In the latter case, the *run* is encoded absolutely. Note that the *run* always refers to the maximum between the absolute displacements in the x and y directions. In both Fig. 6 and 7, vector lengths are proportional to the probability of the corresponding symbol. For both the *angle* and the *run* components, no contextual information is available when there too few reference boundary or control points in the reference window.

The *run* and *angle* components are grouped together for the purpose of forming one symbol. In addition to taking advantage of their possible correlation, doing so avoids inefficiency associated with symbols having an odd *run* and an angle, which is not a multiple of 45^0 - an impossible situation. Probability distributions associated with this aggregate symbol are discussed in Sec. VII.

In addition to *angle* and *run* symbols, there are also several symbols in the encoder alphabet corresponding to tracking the reference contour, i.e., the previously reconstructed boundary, for a number of pixels. These symbols are very useful when at least a part of the boundary is approximated well through motion compensation, particularly at low bit rates (large allowable distortion). With these symbols, henceforth referred to as tracking symbols, the encoder has an extra degree of freedom in approximating large stretches of the original boundary. When a tracking symbol n ($n \in 8, 15, 22, 29, 36, 43, 50$) is transmitted, special care must be taken in re-defining a segment (Eq. 1). Figure 8A shows an instance where the control point (p_{u+1}) is encoded with a tracking symbol, preceded by a non-tracking symbol. In this case, the reference boundary (shown with a dashed line) is being followed for n pixels, starting at the midpoint between p_{u-1} and p_u, the end of the previously encoded segment, and terminating at p_{u+1}. In Fig. 8B, however, both p_u and p_{u+1} are encoded with tracking symbols, which is why the current segment runs from p_u to p_{u+1}. In both cases, a following non-tracking symbol would define a segment, a straight line, between p_{u+1} and p_{u+2}. Note that in order ensure that the approximation results in a closed contour, we restrict the starting pixel of a tracking segment to be within 1 pixel from the end point of the previous spline segment. The last pixel of the tracking segment, while being on the reference boundary, must also be in the admissible control point band of the current contour.

As a result, the symbol stream generated by the encoder, has the following structure. For each consecutive control point, a symbol must be transmitted first indicating whether the tracking mode is used. If yes, it is followed by the corresponding tracking symbol. Otherwise, based on the four possible outcomes of the testing for the presence of the *run* and *angle* contexts, a joint *run, angle* VLC table is used to encode the control point location. Each of these 5 VLC tables can be optimized with the procedure described in Sec. VI.

Having established the *angle* and *run* encoding scheme, we define the total object rate in terms of constituent segment rates. If $r(p_{u-1}, p_u, p_{u+1})$ denotes the segment rate for representing p_{u+1} given control points p_{u-1}, p_u, then the total rate is given by

$$R(p_0, \ldots, p_{N_P-1}) = \sum_{u=0}^{N_p-1} r(p_{u-1}, p_u, p_{u+1}). \tag{4}$$

Note that $r(p_{u-1}, p_u, p_{u+1})$ implicitly assumes the knowledge of the context at point p_u.

Regardless of the context, the first control point location is encoded absolutely and that cost together with the cost of sending a global motion vector, constitutes an overhead outside the realm of the ORD optimization described in Sec. V.

B. Distortion

Spline segment distortions need to be defined in order to evaluate the total boundary distortion. This is done by first associating segments of the approximating curve with segments of the original boundary, as shown in Fig. 9. Here the midpoints of the line segments (p_{u-1}, p_u) and (p_{u+1}, p_u), l and m, respectively, are associated with the points of the boundary closest to them, l' and m'. That is, the segment of the original boundary (l', m') is approximated by the spline segment (l, m).

Once the correspondence problem between the spline segment endpoints and the original boundary is solved, there are many ways in which the segment distortion can be defined. For example, one could use the maximum operator giving the largest distance among any boundary point of the segment and its closest spline counterpart [9]. Another approach is to evaluate the area between the two segments. From the various ways to measure distortion we utilize in this paper the following additive distortion metric per frame, which has also been used in MPEG-4 to evaluate performance of competing algorithms:

$$D_{MPEG4} = \frac{\text{number of pixels in error}}{\text{number of interior pixels}}, \tag{5}$$

where a pixel is said to be in error if it belongs to the interior of the original object and the exterior of the approximating object, or vise-versa.

The spline segment distortion $d(p_{u-1}, p_u, p_{u+1})$, shown in Fig. (9), is computed by counting the number of pixels in error (hollow circles on the figure). Note that this requires quantizing the continuous spline to fit the pixel grid of the image. Special care is taken when associating a spline segment to a segment of the original boundary to ensure that the starting boundary pixel of the next segment coincides with the last boundary pixel of the current segment and that error pixels on the border line between m and m' are not counted twice when computing the next segment distortion. Based on the segment distortions, the total boundary distortion is therefore defined by

$$D(p_0, \ldots, p_{N_P-1}) = \sum_{u=0}^{N_p} d(p_{u-1}, p_u, p_{u+1}), \tag{6}$$

where N_p is the number of control points and $p_{-1} = p_{N_p+1} = p_{N_p} = p_0$. The last equality ensures that an approximation to a closed contour is also closed and simplifies implementation. It is mentioned here that other additive distortion metrics can be used [2], [7], [9].

V. Determining the Optimal Solution

Within the confines of the chosen code structure, we seek an ordered sequence of control points p_i, and their number N_P, which is the solution to:

$$\min_{p_0,\ldots,p_{N_P-1}} D(p_0,\ldots,p_{N_P-1}), \quad \textit{subject to}: \qquad R(p_0,\ldots,p_{N_P-1}) \leq R_{max}, \tag{7}$$

We convert the above constrained minimization problem into an unconstrained one by forming the Lagrangian

$$J_\lambda(p_0,\ldots,p_{N_P-1}) = D(p_0,\ldots,p_{N_P-1}) + \lambda \cdot R(p_0,\ldots,p_{N_P-1}), \tag{8}$$

where for any choice of the multiplier λ, J_λ is the cost function to be minimized. This cost function is expressed as a sum of incremental spline segment costs defined as,

$$w(p_{u-1},p_u,p_{u+1}) = d(p_{u-1},p_u,p_{u+1}) + \lambda \cdot r(p_{u-1},p_u,p_{u+1}). \tag{9}$$

The optimal set of control points $(p_0^*,\ldots,p_{N_P-1}^*)$ is then found by casting the problem as a shortest path in a Directed Acyclic Graph (DAG) with control points playing the role of vertices and incremental costs $w()$ serving as edge weights [2]. Dynamic Programming (DP) is employed to find the shortest path in the DAG for a fixed rate-distortion tradeoff λ. We employ a Bezier curve search [7] in order to arrive at λ^*, the multiplier resulting in the total rate closest to the target rate of R_{max}, in very few iterations.

VI. VLC Optimization

Clearly our claim of optimality is contingent on the chosen code structure, the motion compensation scheme, the width of the control point band, and, to a great extent, on the VLC tables. In [4] it was shown that the rate-distortion efficiency of an ORD optimal solution to the problem of intra mode contour approximation is sensitive to the VLC tables for the *run* and *angle* parameters. Hence, operational optimality of the solution could only be claimed in the following sense:

$$\{p_0^*,\ldots,p_{N_P-1}^*\} = arg[\min_{p_0,\ldots,p_{N_P-1}}[D(p_0,\ldots,p_{N_P-1}) + \lambda \cdot R(p_0,\ldots,p_{N_P-1})|VLC]] \tag{10}$$

In this paper we modify the iterative procedure proposed in [6], [4] to remove the conditioning of the ORD optimal solution on an ad-hoc VLC. As a result of its application, the solution to the following optimization problem is found

$$\{p_0^*, \ldots, p_{N_P-1}^*\} = arg \min_{p_0,\ldots,p_{N_P-1};f\in F} [D^*(\cdot) + \lambda \cdot R^*(\cdot)], \tag{11}$$

where f is a member of the family of context-conditioned parameter probability mass functions F. Hence the shape approximation and the parameter probability model are found jointly and ORD locally optimized.

In the beginning of the iterative process, depicted in Fig. 10, the encoder compresses a sequence of binary frames in the inter mode with a fixed rate-distortion tradeoff λ and an initial probability mass function for (*angle, run* | *context*). Having encoded the input sequence at iteration k, based on the probability mass function $f^k()$, we use the frequency of the output symbols to compute $f^{k+1}()$.

Unlike the intra mode, however, here the symbols representing objects in different frames are correlated. This means that while for the current object we optimize control point locations to achieve the best possible R-D efficiency, the effect on the next frame is not considered. Thus, we assume that even though temporal correlations between contours c_{i-1} and c_i in frames i-1 and i, respectively, exist, the extent to which these correlations can be capitalized upon is independent of a particular approximation of c_{i-1}. Besides being reasonably accurate, especially for noisy contours, this assumptions does not require exhaustive search and makes the optimization procedure tractable. It also hold true in the experiments, discussed in Sec. VII.

In the intra mode, output symbol frequencies at iteration k serve as the basis for the VLC table at iteration $k+1$. Since the sequence of selected control points at iteration k is available to the encoder at iteration $k+1$, with the VLCs derived from that sequence, the encoder can, at the very least, select the same path and, thus, incur a lower or equal Lagrangian cost. The lower or equal cost is a consequence of the VLC tables optimized for that particular path. Therefore, the cost C in Fig. 10 is a non-increasing function of k.

In the inter mode, on the other hand, convergence can not be proved in general. Nevertheless, because of the assumption stated above, we do expect the cost function to monotonically decrease with iterations. However, for the inter mode, the procedure for updating the VLC tables has to be modified to ensure the closure of all contours. To avoid situations when certain $run, angle$ symbols, occurring with a 0 frequency at iteration k and, therefore, not usable at iteration $k+1$, but are required to trace a closed path, we modify the original procedure. The probability distribution f_{k+1}, driving the encoder at iteration $k+1$, is obtained by blending f_k and the distribution inferred from the obtained solution $f_k^{solution}$ as follows:

$$f_{k+1} = \beta \cdot f_k^{solution} + (1-\beta) \cdot f_k, \tag{12}$$

where the blending parameter β was set to 0.75.

The iterative process depicted in Fig. 10 stops when the cost improvement is less than ϵ, at which point the symbols are arithmetically encoded and sent to the decoded together with the overhead of the two probability mass functions.

VII. Results

Figure 11 shows the ORD curves of the proposed algorithm for the SIF sequence "kids". The distortion axis d_n represents the average of the D_{MPEG4}'s defined in Eq. (5) for one frame, over 100 frames. As the figure demonstrates, our result compares favorably with both the baseline [3] and the vertex-based [5] algorithms in the inter mode across most of the range of bitrates. In the very low distortion region ($d_n \leq 0.006$) of operation, however, the proposed algorithm requires more bits than both the baseline and the vertex-based methods. This is due to the fact that for near-lossless boundary encoding the chosen code structure (direction plus run) is inefficient.

In this implementation, 16 directions were allowed in the case the context is present. Encoded differentially with respect to the angle context, they correspond to 16 out of 24 conditional symbols for the direction component. The other 8 symbols are used when a context is not present. The run component was represented by 25 symbols, with only 5 symbols (corresponding to runs of 1, 2, 4, 8, and 12) used for any given context. Additionally, 7 symbols were used for the tracking mode, representing 7 different lengths (spaced uniformly from 8 to 45) for which a reference contour could be tracked. Bit-rates for the proposed method, reported in Fig. 11, take into account bits for the global motion vectors, searched in a 32×32 window.

A. Initialization of Symbol Probabilities

The iterative scheme for optimizing the VLCs, described in Sec. VI, is locally optimized in the following sense. Let f_T be the parameter probability mass function, with its corresponding cost C_T, at the termination of the iterative process depicted in Fig. 10. Then $f = f_T + f_\epsilon$, another probability mass function separated by a small distance from f_T, will result in a Lagrangian cost $C \geq C_T$ when applied to the optimal encoder under the same rate-distortion tradeoff λ. Since C_T is a local minimum, there is no guarantee that perturbations to f_T larger than f_ϵ will not outperform it in terms of the Lagrangian cost.

To get an idea of how close a local minimum C_T is to the global minimum, we conducted several experiments with different initializations of the probability mass function f_0. Figure 12 shows the corresponding rate-distortion curves.

Figure 13 shows the symbol probability distribution, with the *run* context of 8, after 5 iterations of the process described in Sec. VI. As expected, a sharp peak occurs for the run of 8 and the angle displacement of 0. This indicates that following the context in both the *run* and the *angle* was the most frequently occurring symbol.

VIII. Conclusions

In this work we proposed a novel technique for capitalizing on the between frame correlation of shape information, while achieving an ORD optimal performance. In addition, we have removed the dependency of the ORD optimal solution on a particular VLC used to encode the generated symbols.

Although the proposed algorithm clearly outperforms existing inter mode techniques, its overall improvement in efficiency of shape representation with respect to the intra mode is not comparable to that of texture representation, where temporal correlation is exploited to a much larger degree. We expect that further gains can be made by adaptively adjusting the size of the context window with λ, and, possibly, by using a more sophisticated motion model. Also, a hybrid intra - inter technique, utilizing prediction with respect to both the context and the previously encoded control point, may outperform this approach. Since the use of context couples consecutive frames, the global (across objects) optimality is lost. Hence an approach using different values of λ for the same object in different frames may potentially be more efficient.

References

[1] N. Brady, F. Bossen, and N. Murphy, "Context-Based Arithmetic Encoding of 2D Shape Sequences", *Proc. ICIP97*, pp. I-29-32, 1997.

[2] A. K. Katsaggelos, L. Kondi, F. W. Meier, J. Ostermann, G.M. Schuster, "MPEG-4 and Rate Distortion Based Shape Coding Techniques", *Proc. IEEE*, pp. 1126-1154, June 1998.

[3] S. Lee, *et al.* "Binary Shape Coding using 1-D Distance Values from Baseline", *Proc. ICIP97*, pp. I-508-511, 1997.

[4] G. Melnikov, G. M. Schuster, A. K. Katsaggelos, "Simultaneous Optimal Boundary Encoding and Variable-Length Code Selection", *Proc. ICIP98*, pp. I-256-260, Oct. 1998.

[5] K. J. O'Connell, "Object-adaptive Vertex-Based Shape Coding Method", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, pp. 251-255, Feb. 1997.

[6] D. Saupe, "Optimal Piecewise Linear Image Coding", *Proc. SPIE Conf. on Visual Comm. and Image Proc.*, vol. 3309, pp. 747-760, 1997.

[7] G. M. Schuster and A. K. Katsaggelos, *Rate-Distortion Based Video Compression, Optimal Video frame compression and Object boundary encoding.* Kluwer Academic Press, 1997.

[8] G. M. Schuster and A. K. Katsaggelos, "An optimal polygonal boundary encoding scheme in the rate distortion sense," *IEEE Transactions on Image Processing*, vol. 7, no. 1, pp. 13-26, Jan. 1998.

[9] G. M. Schuster, G. Melnikov, and A. K. Katsaggelos, "Optimal Shape Coding Techniques," *IEEE Signal Processing Magazine*, Nov. 1998.

[10] N. Yamaguchi, T. Ida, and T. Watanabe, "A Binary Shape Coding Method using Modified MMR", *Proc. ICIP97*, pp. I-504-508, 1997.

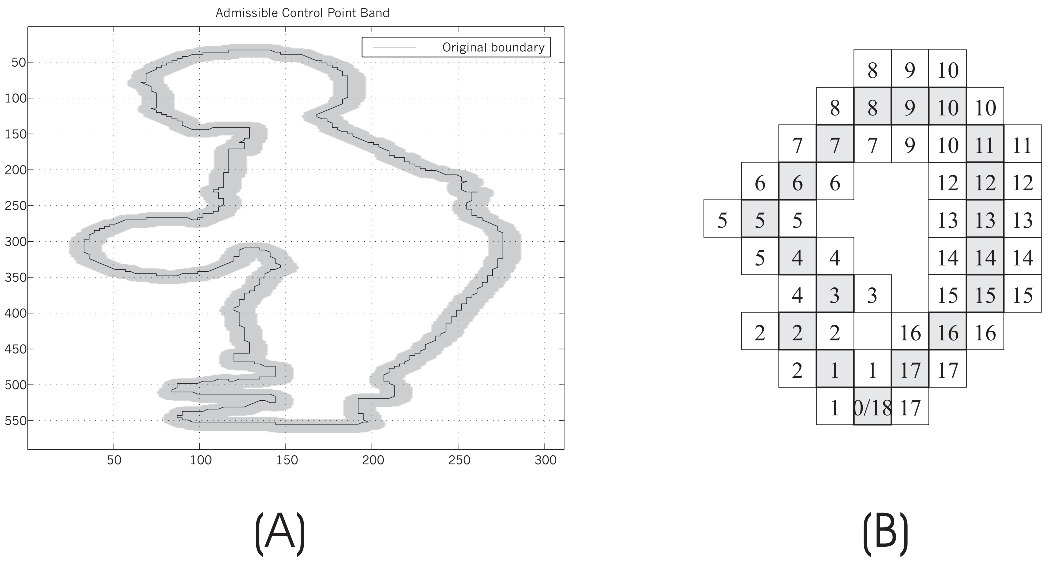


Fig. 1. (A) Admissible control point band; (B) Ordering of admissible control points .

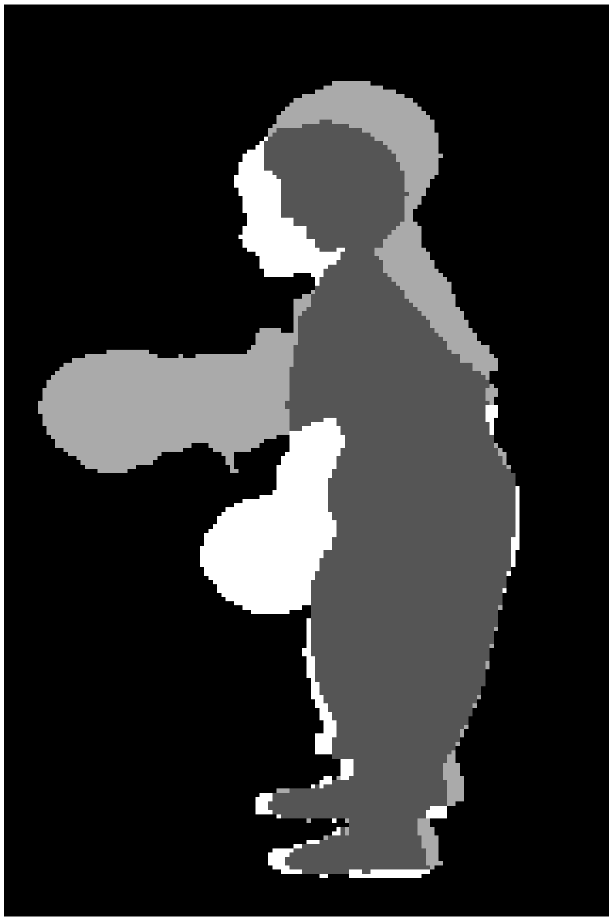


Fig. 2. Original and motion compensated binary frames; dark grey region: overlapping region, white region: error region, object in current frame, background in previous frame, light grey region: error region, background in current frame, object in previous frame. Sources of error: non-rigid motion and the acquisition/segmentation process.

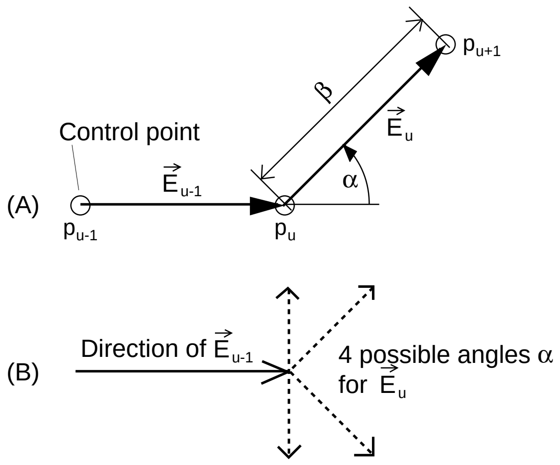


Fig. 3. Encoding of a spline control point.

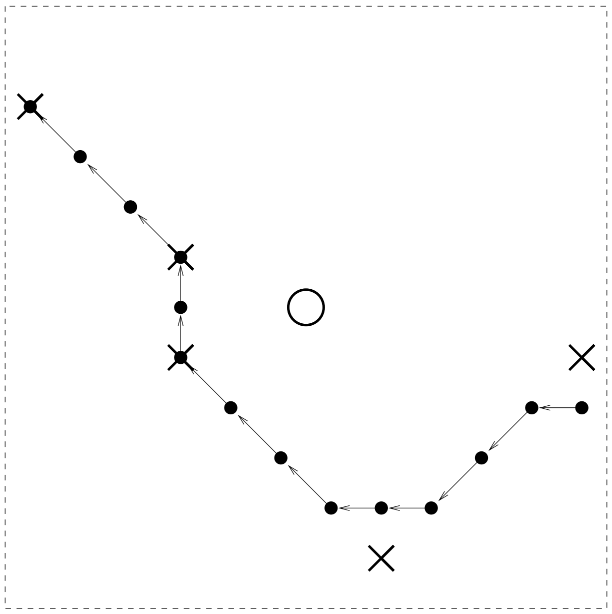


Fig. 4. Control points (X marks) and boundary pixels (circles) in a temporal context window. Context: NW direction, run of 4.

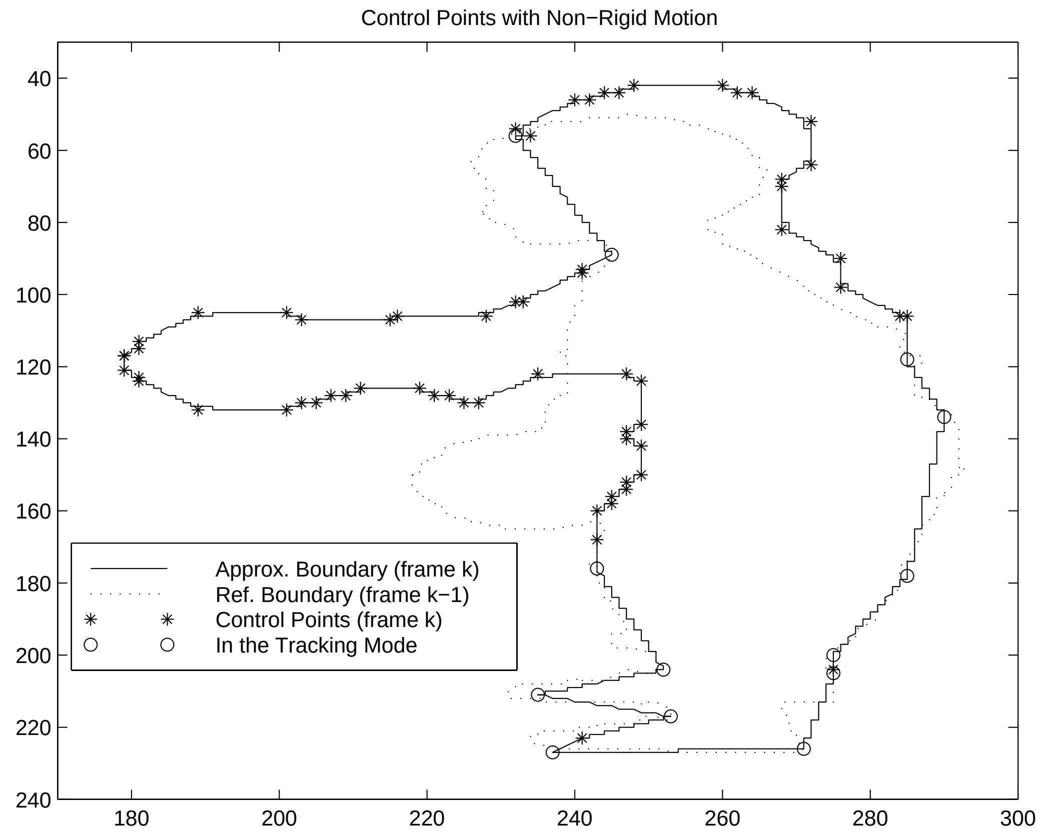


Fig. 5. Control point placement under very low bit rate. Circles correspond to the tracking mode.

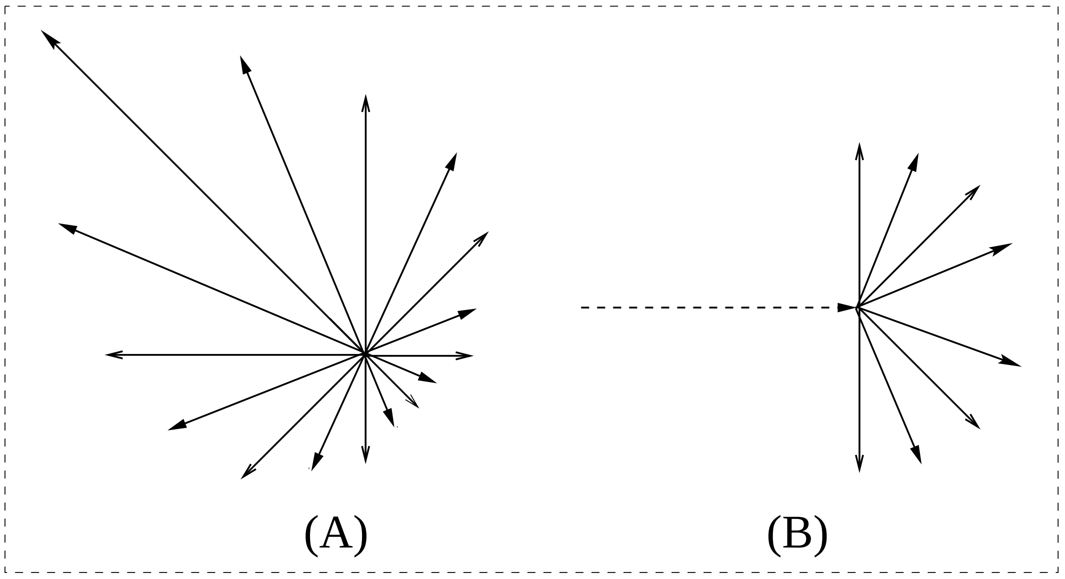


Fig. 6. Typical probability assignment for the direction with NW context (A), with no context (B).

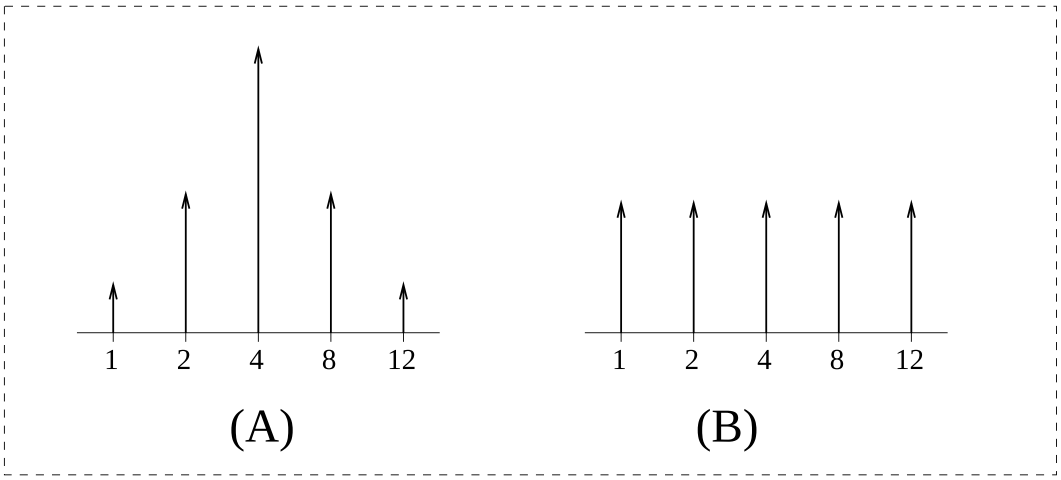


Fig. 7. Typical probability assignment for the run with a context of 4 (A), with no context (B).

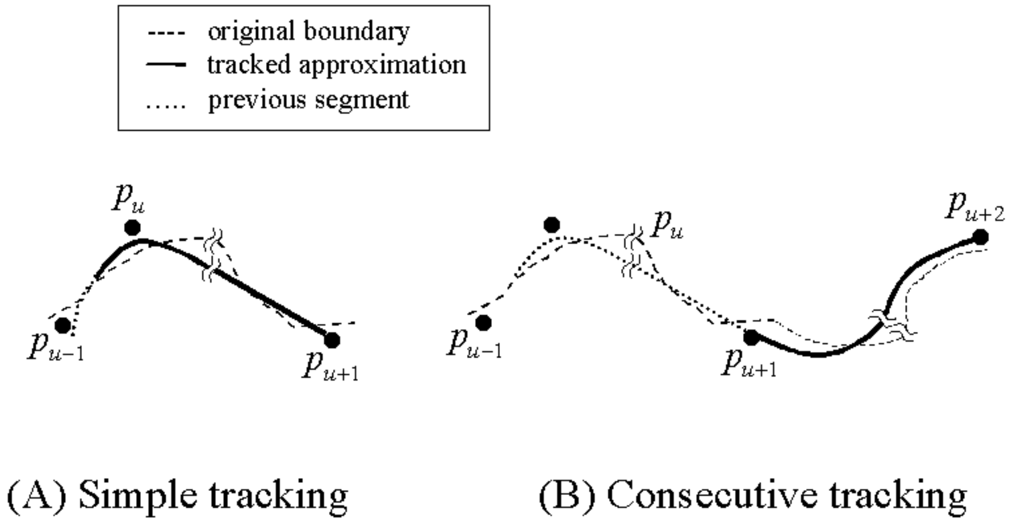


Fig. 8. Encoding p_{u+1} in the tracking mode.

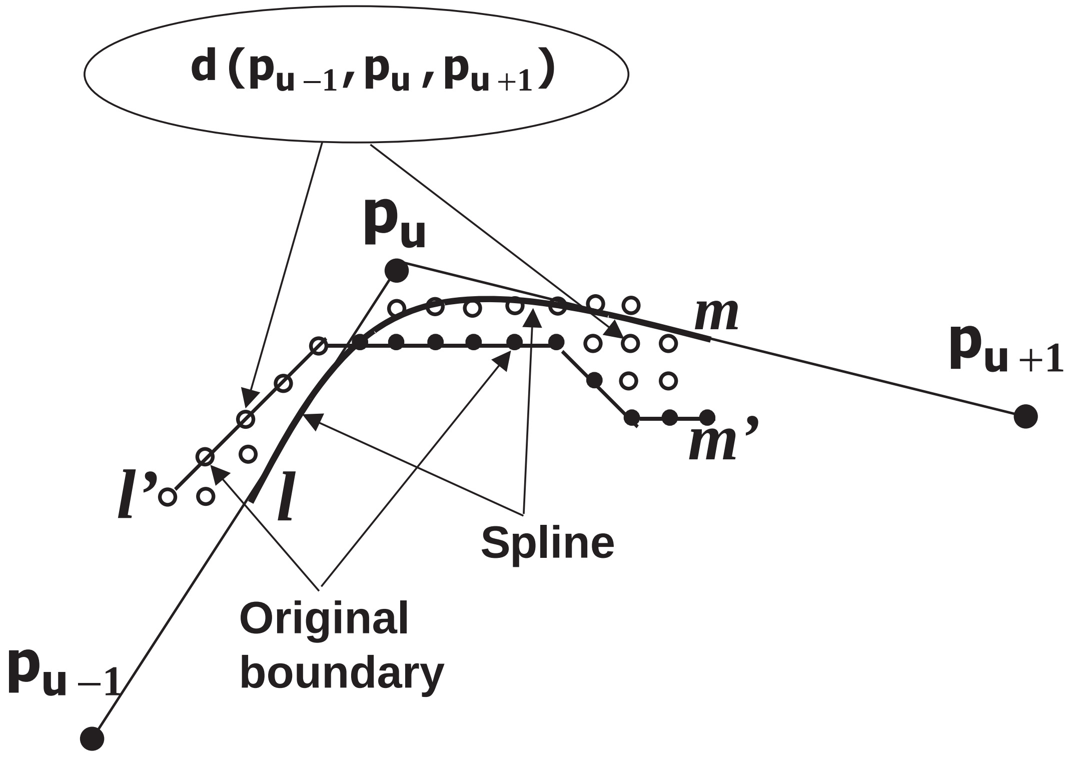
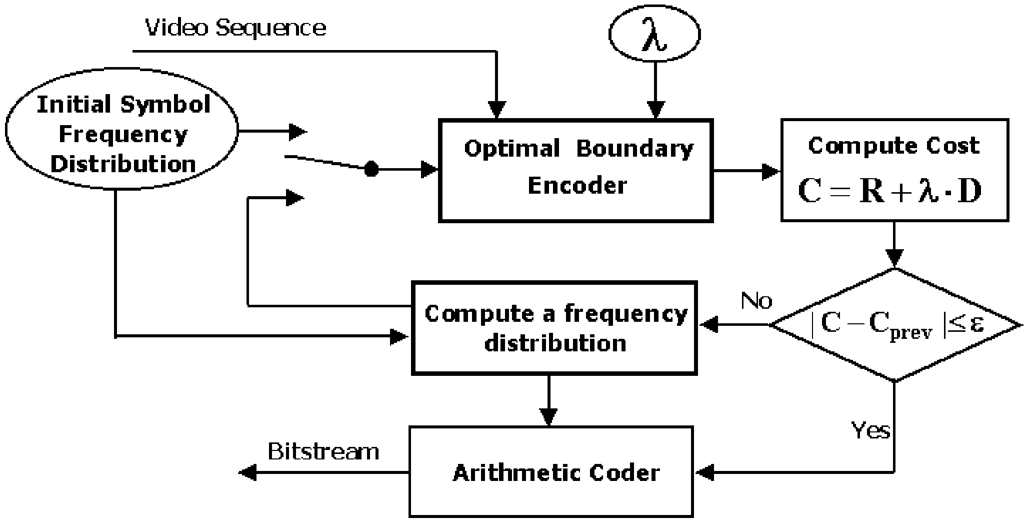


Fig. 9. Area between the original boundary segment and its spline approximation (circles).



Video Sequence
λ
Initial Symbol Frequency Distribution
Optimal Boundary Encoder
Compute Cost C = R + λ · D
Compute a frequency distribution
No
|C − C_prev| ≤ ε
Yes
Bitstream
Arithmetic Coder

Fig. 10. The entropy encoder structure.

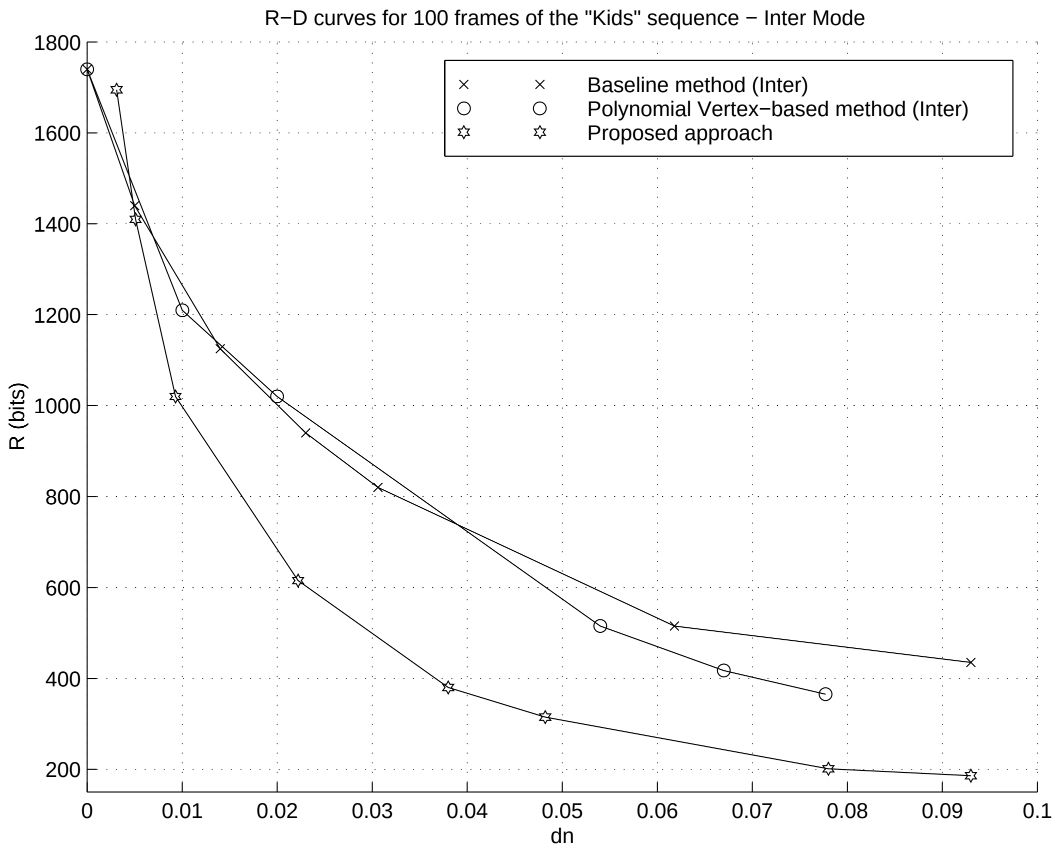


Fig. 11. Rate-Distortion curves.

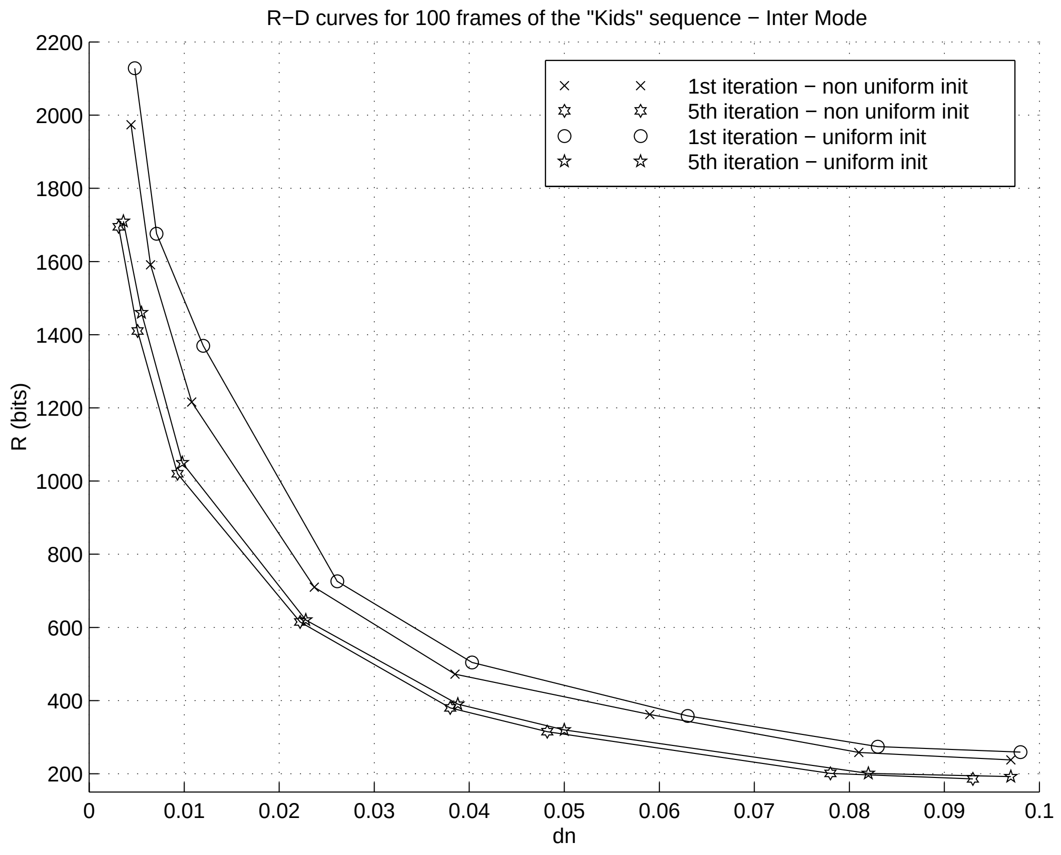


Fig. 12. Rate-Distortion curves for different initializations of f_0.

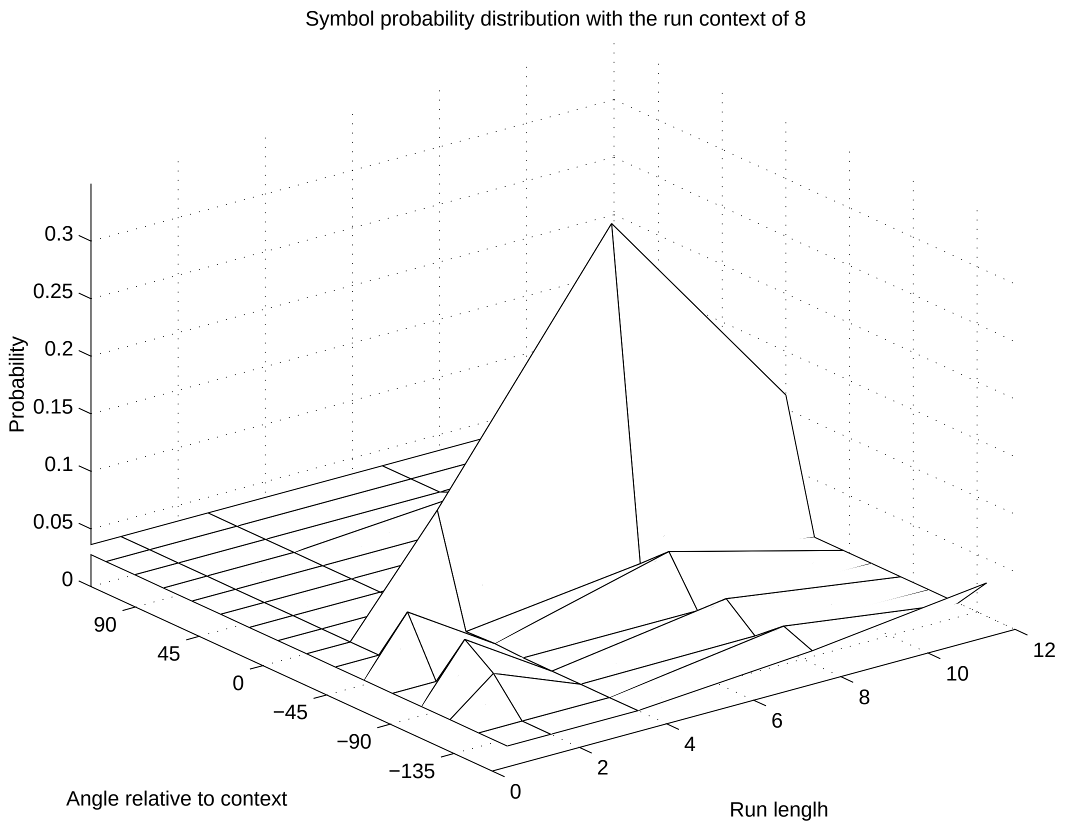


Fig. 13. Probability distribution at convergence for run context of 8.