RATE DISTORTION OPTIMAL CONTOUR COMPRESSION USING CUBIC B-SPLINES

J. Zaletelj[(1)], R. Pecci*, F. Spaan[(2)], A. Hanjalic[(2)], R.L. Lagendijk[(2)]

[(1)] *Faculty of Electrical Engineering*
University of Ljubljana
Ljubljana, Slovenia
janez.zaletelj@fe.uni-lj.si

[(2)]*Information and Communication Theory Group*
Department of Information Technology and Systems
Delft University of Technology
{franks,alan,inald}@ict.its.tudelft.nl

ABSTRACT

Object contours resulting from segmenting images or video frames can be efficiently encoded using B-spline functions. An unsolved problem is how to divide a given contour into segments such that the resulting compression is optimal in rate-distortion sense. In this paper we describe two techniques for finding a close-to-optimal knot assignment. The first technique prunes an accurate B-splines approximation until the desired rate is achieved. The second technique analyzes the curvature function of the original contour to obtain a suboptimal knot assignment. The resulting algorithms are compared experimentally.

I. INTRODUCTION

The representation of object shapes, or *contours*, is a research topic with a long history. Whereas contour descriptors originally aimed at reliable *recognition* and *comparison* of shapes in a variety of robot vision problems, an additional constraint has emerged in the context of object-based image and video *compression* for communications and multimedia. In these applications the *efficiency* of a particular descriptor is also highly important, both in terms of the average bit rate per contour pixel (BPCP) as well as the computational complexity of the encoding and reconstruction process. Though thorough perceptual studies have not been carried out to date, the general feeling is that slightly *lossy* compression is acceptable in compressing object shapes at the advantage of a higher compression factor. Two categories of shape descriptors have been studied primarily, namely [3-7]:

- chain codes and their descendants,
- polynomial functions.

Most of the existing techniques share the drawback that explicit control of the encoded bit rate is not possible. Since compression systems have to be able to trade-off the bit rate and resulting distortion among the various types of information to be encoded for each object in an image or video frame (shape, texture, motion), contour compression methods should also be equipped with a possibility for bit rate control.

In this paper we consider the problem of representing a given contour by a collection of uniform cubic B-splines. The basic idea is that segments of the original contour are approximated by third order polynomial functions (in x and y coordinate). This approximation is such that the individual polynomial functions are connected and smooth. It has been shown in [1,2] that once the original contour is divided into segments, the optimal MSE spline approximation of this contour can be found by solving a set of linear equations. Therefore the problem that we concentrate on in this paper is the following: *"How should the original contour be broken up into segments such that a (close to) rate-distortion optimal contour compression results."* The points where two adjoining segments meet are called *knots*. The optimal determining of the knot locations is not a trivial problem since the possible number of ways to divide a given contour into n segments is enormous. Furthermore, it is difficult to predict the rate of the resulting B-spline approximation if an arbitrary division of the original contour is taken.

In Section II of this paper we first present some background information. Then we propose two techniques for addressing the knot assignment problem. The method described in Section III is based on pruning spline segments from an initially very accurate (high bit rate, low distortion) splines approximation. This elegant approach is purely driven by the desire to obtain a rate-distortion optimal approximation. In Section IV we then continue to propose a more heuristic, but computationally far more efficient method by carefully analyzing the curvature function of the original contour. The two techniques are experimentally compared in Section V.

II. B-SPLINE BACKGROUND

We assume that the original (closed) contour is given as an ordered set of M (simply) 8-connected pixels, denoted by $\{S_1,S_2,...,S_M\}$. This contour is divided into n segments, and each segment $C_i(u)$ is approximated by a uniform cubic B-spline [1,2]:

$$C_i(u) = \begin{bmatrix} x_i(u) \\ y_i(u) \end{bmatrix} = \left[u^3, u^2, u, 1\right]\mathbf{M}\left[P_{i-1}, P_i, P_{i+1}, P_{i+2}\right]^T . \quad (1)$$

Here $\mathbf{M}$ is the matrix with spline basis functions, and $P_k = [P_k(x), P_k(y)]$ refers to the coordinates of the k-th control point of the spline approximation. The term *uniformity* comes here from the fact that in segment i the spline u-parameter runs from $[i\text{-}1,i]$. Uniformity reflects therefore

* Ms. R. Pecci, M.Sc. was an exchange student under the EU Erasmus program from University of Firenze, Italy.

only the parameterization of the spline functions, and not the actual length of the spline segments.

If we now assume that the attribution of the parameter u to the original contour points S_j is known, i.e. each pixel on the original contour has been assigned a u-value, then we can find the n control points P_k such that the MSE distortion between approximating spline and original contour pixels is minimized [1]. Clearly the attribution of the u-parameter – and consequently the division of the original contour into segments – is the crucial step prior to the actual computation of the control points. In the literature, several possible parametrizations can be found, for instance uniform, chord length, centripetal or Foley parametrization [10]. None of these techniques, however, takes the resulting rate and distortion sufficiently into account. In the following two sections we therefore propose alternative methods to break the original contour into segments.

III. B-Spline Pruning

In our first approach [8], we start out with an initial approximation of the original contour using a large number of segments, i.e. a virtually error-free contour approximation. There are maximally as many spline segments as there are contour pixels (M). We have, however, experimentally determined that a sufficiently large number of initial segments n_{ini} is:

$$n_{ini} = \tfrac{1}{2}\left\lfloor \frac{M}{\sqrt{2}} \right\rfloor. \qquad (2)$$

Subsequently, the initial u-parameter attribution is:

$$u_i = u_{i-1} + \frac{\left|S_i - S_{i-1}\right|}{\left|S_M - S_1\right|} n_{ini}\,. \qquad (3)$$

Given this initial assignment, the B-splines approximation can be determined, yielding a set of control points $\{P_1,P_2,...,P_P\}$. The result is a very precise approximation requiring n B-spline segments and n control points. The resulting B-splines functions are continuous in x and y. Although the final result of the contour compression has to be a discrete contour, we do not convert intermediate B-spline approximations to discrete contours but keep them as continuous functions instead, such that we can more accurately measure the distortion between original and compressed contour.

We then eliminate co-linear control points in the initial approximation. If N>6 control points are approximately on a straight line, then N-6 intermediate control points can be eliminated without changing the B-spline approximation. Removing co-linear control points does not change the distortion but does decrease the number of spline segments – and therefore the bit rate.

The bit rate of the resulting initial B-splines approximation is most likely too large. To achieve the desired bit rate, the set of approximating B-splines is pruned iteratively, in the sense that the number of spline segments is decreased iteratively by merging two adjoining segments. For each pair of adjoining segments the decrease in bit rate ΔR and increase in distortion ΔD is calculated if the segment pair would be replaced by a single segment. Then those two adjoining segment are merged for which the ratio $|\Delta R/\Delta D|$ takes on the largest value. The rate calculation is based on first order DPCM combined with Huffman encoding of the control points, and the distortion is the quadratic (MSE) distance between original and approximating contour. This pruning process is repeated iteratively until the desired rate has been achieved.

The above mentioned criterion for merging adjoining segments in combination with the convex nature of the rate-distortion curve of the splines approximation guarantees a solution that is optimal in rate-distortion sense. However, the rate-distortion optimality also requires that after merging two segments, a new u-parameter attribution and a new least-squares B-splines approximation of the original contour should be calculated. Consequently, after each pruning step the effect of merging adjoining segments should be recalculated entirely. Since these steps are the most computationally expensive parts of the algorithm, we introduce two suboptimal strategies which reduce the complexity significantly:

- in a single iteration multiple segments (K) are pruned in the order of decreasing $|\Delta R/\Delta D|$ ratio,
- after pruning one of more segments, the B-spline approximation is only locally recalculated.

Every now and then a "global" recalculation of the entire B-spline approximation has to be carried out to correct the "drift" effects of the local recalculations of the B-spline. The resulting iterative scheme is shown in Figure 1.

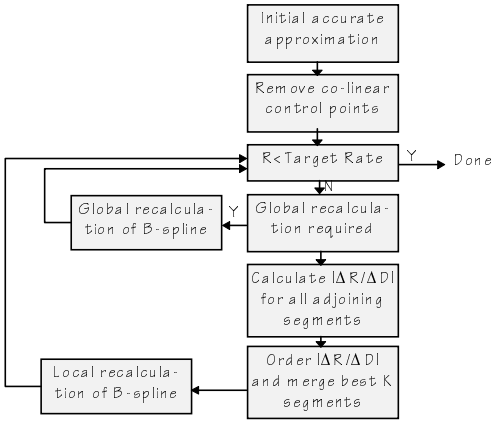

Figure 1: Block-diagram of iterative pruning of spline segments.

The frequency of the global recalculation and the number of segments that can be pruned in a single iteration may greatly influence the degree of suboptimality. We have experimentally studied the influence of these parameters on a variety of contours [8]. As an example, Figure 2 shows (for a case with a prescribed distortion) the bit rate as a function of the number of prunes per iteration and as a function of the periodicity of global recalculation. We see that below approximately 10-15 prunes per iteration and a

periodicity of smaller than $n/10$, the resulting bit rate fluctuates randomly. This indicates that for those parameter choices the proposed suboptimal strategies do not compromise the optimality of the solution significantly.

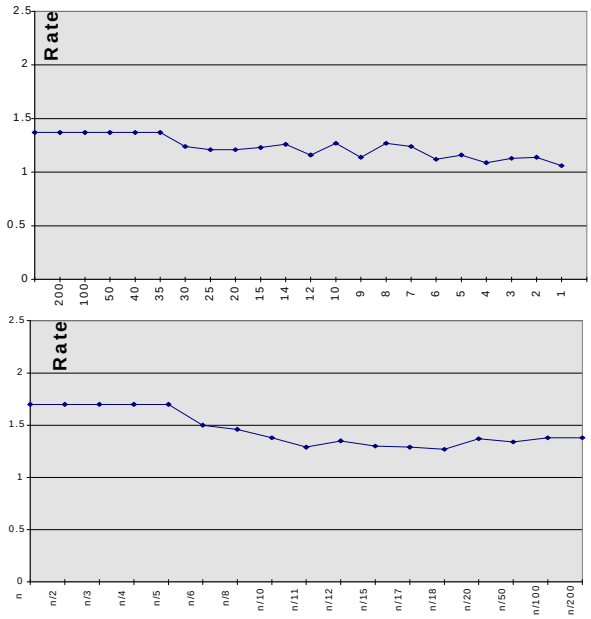

Figure 2: Illustration of the effects of periodicity of global optimization (top) and number of prunes per iteration (bottom, n is the current number of spline segments) on the compression performance.

IV. Analyzing the Curvature Function

If one analyzes the knot assignments resulting from the pruning approach described in the previous section, it shows that

- many of the knots are located closely to or at positions on the contour where the curvature has extremes,
- usually each segment contains one curvature extreme.

Figure 3 illustrates these effects. This observation leads to the idea of making the knot assignments dependent on the curvature of the original contour right from the beginning. We will initially position a knot at the location of each curvature extreme. We first calculate the curvature function [9] of the original contour, defined by:

$$\kappa(t) = \frac{x_t(t)y_{tt}(t) - x_{tt}(t)y_t(t)}{(x_t^2(t) + y_t^2(t))^{3/2}}. \qquad (4)$$

The curve of which the curvature is calculated, is assumed to be equidistantly resampled yielding $x(t)$ and $y(t)$. The parameter t indicates arc length along the contour. The subscripts in (4) refer to their first and second derivatives. As mentioned above – depending on the desired bit rate – the curvature should have a few curvature extremes (few knots, few segments, and therefore a low bit rate) or many extremes (high bit rate). To control the bit rate, we need to control the number of extremes. We do this by smoothing $x(t)$ and $y(t)$ using a Gaussian kernel $g_t(t,\sigma)$ of variable width σ.

The parameter σ now controls the bit rate. In (4), the derivatives are subsequently replaced by their smoothed versions, for instance $x_t(t)=x(t)\otimes g_t(t,\sigma)$. The appropriate value of σ is determined as follows. Starting with the initial value of σ, we calculate the derivative of the curvature function $\kappa_\sigma(t)$. We count the number of zeros of $\kappa_\sigma(t)$, and if it is larger than the desired number of segments, we increase σ by σ_{step}, otherwise we decrease σ by σ_{step}. This (global) search procedure is repeated until the smallest σ, which results in the desired number of curvature extremes, is achieved. This algorithm is computationally efficient, because it requires simple 1-D convolutions and typically less than 10 iterations, and it ensures, that the resulting curvature function contains only the desired number of most prominent features.

For a calculated value of σ and (smoothed) curvature function, the initial division of the contour into segments assigns a knot to each curvature extreme. Then, for each initial segment, its length and the integral of $|\kappa(t)|$ is determined. Based on these two measures segments are nominated for merging or splitting. The splitting and merging procedure uses only curvature information, and the following two additional constraints:

- each resulting segment may have no more than one curvature zero crossing,
- the quotient of the lengths of the adjoining segments should be smaller than a given threshold.

If the integral of $|\kappa(t)|$ and length of the segment i exceed the average values, then the segment is nominated for splitting, and the measure of adequacy sa_i is calculated:

$$sa_i = \frac{arclength(i)}{\frac{1}{n}\sum_{\forall i} arclength(i)} \frac{\int_{\text{segment } i} |\kappa(t)|dt}{\frac{1}{n}\sum_{\forall i} \int_{\text{segment } i} |\kappa(t)|dt}. \qquad (5)$$

If the two adjoining segments i, j do not include any curvature zero crossings, they are nominated for merging, and the measure of adequacy ma_i is calculated:

$$ma_i = \frac{\left|\max(\kappa_i(t),\kappa_j(t)) - \min(\kappa_i(t),\kappa_j(t))\right|}{\max_n\left(\left|\kappa_n(t)\right|\right)} + \frac{\frac{1}{2}\left|\max(\kappa_i(t),\kappa_j(t)) + \min(\kappa_i(t),\kappa_j(t))\right|}{\max_n\left(\left|\kappa_n(t)\right|\right)} \qquad (6)$$

The merging and splitting lists are sorted according to the ma_i and sa_i, but the actual number of performed mergings and splittings depends on the desired number of segments (bit rate). The above criteria are clearly much more heuristic than the ones in Section III, but the entire process is computationally far less expensive.

The final step of the algorithm is B-spline control point calculation. We assign a parameter (u) to each point of the original contour, and the control points are calculated by solving the matrix equation (using QR decomposition). The actual distortion and bit rate of the approximation are calculated, and if they do not meet the requirements, the algorithm starts again, using adjusted input parameter (number of segments).

V. EXPERIMENTAL COMPARISON

We have experimentally compared the two proposed methods for knot assignment. Figure 3(a) illustrates the final location of the knots (vertical lines) of the *pruning* approach drawn in the curvature plot of the *splines* approximation, while Figure 3(b) shows the resulting knot assignment of the *curvature analysis* approach drawn in the curvature plot of the *smoothed original* contour. Figure 4 shows the reconstructed splines approximation. Clearly there is a lot of correspondence between results. Computationally, however, there is a difference of a factor 50 to 150 in favor of the curvature analysis approach. Finally, Figure 5 shows the realized bit rate versus (MSE) distortion for the two proposed techniques.

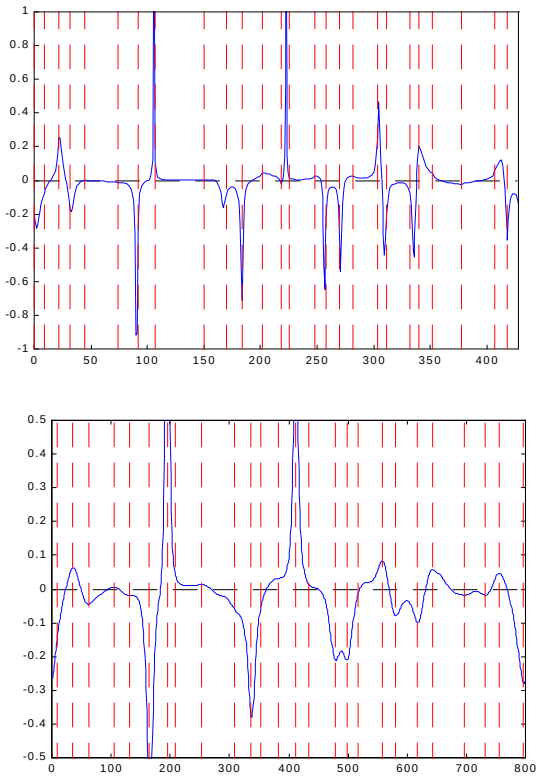

Figure 3: Curvature function based on (top) splines approximation, (bottom) smoothed original contour, and location of knots for (top) pruning and (bottom) curvature analysis approach.

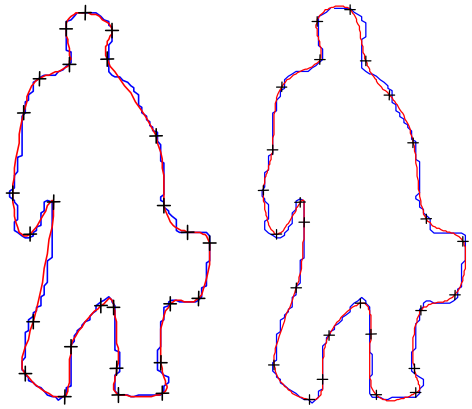

Figure 4: Spline approximation based on knot assignment by segment pruning (left) and curvature analysis (right). Knot locations are indicated by + signs.

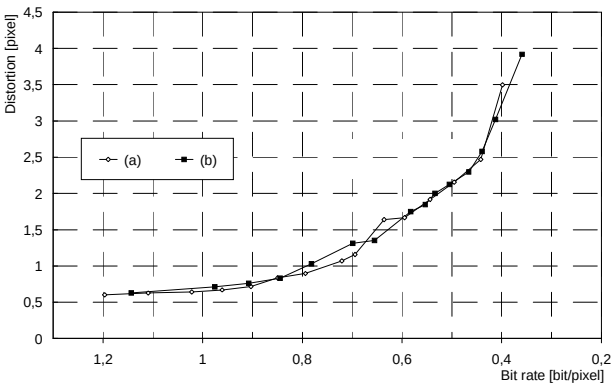

Figure 5: R-D performance of the two proposed methods, (a) B-spline pruning, (b) curvature analysis approach.

REFERENCES

[1] F. Lu and E.E. Milios, "Optimal splines fitting to planar shape", *Signal Processing*, vol. 37, pp. 129-140, 1994.

[2] F. Cohen, Z. Huang , Z. Yang, "Invariant matching and identification of curves using B-splines curve representation" *IEEE Trans. Image Proc.* 4 (1), 1995.

[3] P.Brigger, M. Kunt, "Morphological shape repre-sentation for very low bit-rate video coding" In *Signal Processing: Image Comm.* 7, pp. 297-311, 1995.

[4] M. Eden, M. Kocher "On the performance of a contour coding algorithm in the context of image coding.Part I: Contour segment coding". *Signal Processing* 8, pp.381-386, 1985.

[5] F.W. Meier, G.M. Shuster, A.K. Katsaggelos ,"An efficient boundary encoding scheme using B-spline curves which is optimal in the rate distortion sense". *Proc. of VCIP*, 1997.

[6] R.L. Lagendijk, J. Biemond, C.P. Quist, "Low bit rate video coding for mobile multimedia communications", *Proc. of the 8-th European Signal Processing Conference EUSIPCO-96*, Trieste, pp. 435-438, 1996.

[7] S. Biswas and S.K. Pal. "Approximate coding of digital contours". In *IEEE Trans. On Systems, Man, and Cybernetics*, SMC-18(6), pp. 1056-1066, 1988.

[8] R. Pecci, *A rate distortion optimal contour approximation using cubic B-splines*, M.Sc. thesis http://www-it.et.tudelft.nl/itbibliography/reports/1997/msc-theses/pecci_thesis.zip

[9] F. Mokhtarian, A.K. Mackworth, "A theory of multiscale, curvature-based shape representation for planar curves", *IEEE Trans. PAMI*, vol.14, no.8, pp.789-805.

[10] G. Farin, "Curves and surfaces for computer aided geometric design, a practical guide", *Academic Press*, Inc., New York, 1994.