An efficient boundary encoding scheme using B-Spline Curves which is optimal in the rate-distortion sense

Fabian W. Meier, Guido M. Schuster, Aggelos K. Katsaggelos

Northwestern University, Department of Electrical and Computer Engineering, Evanston IL, U.S.A.
Email: FabianMeier@nwu.edu, gschuste@usr.com, aggk@ece.nwu.edu

Abstract

A major problem in object oriented video coding is the efficient encoding of object boundaries. In this paper, we present a efficient method for the lossy encoding of object boundaries which are given as 8-connect chain codes. We approximate the boundary by a second order B-spline curve and consider the problem of finding the curve with the lowest bit rate for a given distortion. This scheme is optimal, efficient and offers complete control over the trade-off between bit-rate and distortion. The scheme is an extension of the approaches in [][], where we used polygons to approximate a boundary. The main reason for using curves rather than polygons is that curves a have more natural appereance than polygons. Finally we present results of the proposed schemes using objects from "....." sequences.

1 Introduction

This research is motivated by object oriented video coding [1, 2], where encoding of planar curves is an important problem, but the developed algorithms can also be used for other applications, such as CAD, object recognition, object oriented video coding, etc. The objectiv of object oriented coding is to encode video sequences with a very low bit rate. Efforts are currently being undertaken to incorporate basic standards for object oriented video coding in the MPEG standard []. The basic approach of object oriented video coding is to analyze an image sequence, synthesize the images with objects described by shape, motion and color, and encode this information efficiently. Our paper presents an scheme for efficient encoding of object shapes, where we convert a two-dimensional object boundary into a lossy approximation with a lower bit rate than the rate of the original boundary. The efficiency of a boundary encoding scheme can be measured either with the absolute rate in bits or with a relative measure of the rate in bits divided by the number of boundary points. In this paper we are using the latter relative measure e with the unit bits per boundary point.

Freeman [3] originally proposed the use of chain coding for boundary quantization and lossless boundary encoding, which has attracted considerable attention over the last thirty years [4, 5, 6, 7, 8]. The most common chain code is the 8-connect chain code which is based on a rectangular grid superimposed on a planar curve. The curve is quantized using the grid intersection scheme [3] and the quantized curve is represented using a string of increments. Since the planar curve is assumed to be continuous, the increments between grid points are limited to the 8 grid neighbors, and hence an increment can be represented by 3 bits. There have been many extensions to this basic scheme such as the generalized chain codes [4], where the coding efficiency has been improved by using links of different length and different angular resolution. In [7] a scheme is presented which utilizes patterns in a chain code string to increase the coding efficiency and in [8] differential chain codes are presented, which employ the statistical dependency between successive links. There has also been interest in the theoretical performance of chain codes. In [5] the performance of different quantization schemes is compared, whereas in [6] the rate distortion characteristics of certain chain codes are studied. In this paper, we are not concerned with the quantization of the continuous curve, since we assume that the object boundaries are given with pixel accuracy.

In [9] B-spline curves are used to approximate a boundary. An optimization procedure is formulated for finding the optimal locations of the control points by minimizing the mean squared error between the boundary and the approximation. This is an appropriate objective when the smoothing of the boundary is the main problem. When, however, the resulting control points need to be encoded, the tradeoff between the encoding cost and the resulting distortion needs to be considered. By selecting the mean squared error as the distortion measure and allowing for the location of the control points to be anywhere on the plane, the resulting optimization problem is continuous and convex and can be solved easily. In order to encode the positions of the resulting control points efficiently, however, one needs to quantize them, and therefore the optimality of the solution is lost. It is well known that the optimal solution to a discrete optimization problem (quantized locations) does not have to be close to the solution of the corresponding continuous problem. In a similar approach [10] a new spline, the quasi-orthogonal Q-spline curve, was defined and used for least square fitting.

Average rates of 0.88 bits per boundary point were achieved in [11] with third order B-spline curves to approximate boundaries with a maximum distortion of 1.0 pixel inside and 2.0 pixel outside the boundary. The basic idea of this approach is to start with a initial curve approximation and then to add and remove control points in an iterative algorithm until no further reduction of the rate can be achieved. The results, however, are not unique nor optimal and depend very much on the initial curve.

A boundary with many straight lines or lines with a very small curvature could be encoded more efficiently with a 8-connect chain code combined with run-length encoding. This is the idea of some preprocessing algorithms [12] which are used to "straighten" the boundary to a shape that can be encoded with a lower rate. The problem with such an approach is that there is at best an indirect control over the resulting rate distortion tradeoff.

In this paper, we present algorithms where the preprocessing step and a modified 8-connect chain code/run-length encoding are combined into an optimal lossy segmentation encoding scheme. Note that this is achieved in an optimal fashion, with a complete control over the tradeoff between distortion and bit rate, resulting in an efficient encoding scheme.

In section 2 we define the problem and introduce the required notation and focus on the distortion measure which is based on the maximum operator. The B-spline curve is briefly reviewed in section 3. The definiton of the set of admissible locations for control points for the spline approximation is discussed in section 4. In section 5 we consider the problem of finding the B-Spline curve which requires the smallest bit rate for a given distortion. We solve this problem by introducing a scheme which is based on a shortest path algorithm for a weighted directed acyclic graph. We then consider the dual problem, that of finding the polygon with the smallest distortion for a given bit rate. In section 7 we introduce a control point encoding scheme. Finally in section 8 we present results of the proposed algorithm.

2 Problem Formulation

The main idea behind the proposed approach is to approximate a given boundary by a second order B-spline curve, and to encode the control points that define the B-spline curve.

The following notation will be used. Let $B = \{b_0, \ldots, b_{N_B-1}\}$ denote the connected boundary which is an ordered set, where b_j is the j-th point of B and N_B is the total number of points in B. Note that in the case of a closed boundary, $b_0 = b_{N_B-1}$. Let $P = \{p_0, \ldots, p_{N_P+1}\}$ denote the set of control points of the B-Spline Q, which is also an ordered set, with N_P the total number of curve segments. Note that every curve segment shates control points with its neighboring curve segments. Every B-spline curve segment Q_k is defined by three control points p_{k-1}, p_k, p_{k+1}. Since P is an ordered set, the ordering rule and the set of control points uniquely define the curve.

We assume that the control points of the curve are encoded differentially which is an efficient method for natural boundaries since the location of the current control point is strongly correlated with the location of the previous one. We denote the required bit rate for the differential encoding of control point p_k given control point p_{k-1} by $r(p_{k-1}, p_k)$. Hence the bit rate $R(p_0, \ldots, p_{N_P-1})$ for the entire polygon is,

$$R(p_0, \ldots, p_{N_P+1}) = \sum_{k=0}^{N_P+1} r(p_{k-1}, p_k), \qquad (1)$$

where $r(p_{-1}, p_0)$ is set equal to the number of bits needed to encode the absolute position of the first vertex. However, since the bit size of the encoded absolute location depends on the size of the image plane, we neglect the rate of the absolute position when we calculate the coding efficiency e. For a closed boundary the first control point is identical to the last one, the rate $r(p_{N_P-2}, p_{N_P-1})$ is set to zero since the last control point does not need to be encoded. Note that the rate $r(p_{k-1}, p_k)$ depends on the specific control point encoding scheme.

In general the B-spline curve which is used to approximate the boundary could be permitted to place its control points anywhere on the image plane. In order to restrict the search for relevant control points we define a A as a set of admissible control points. In section 4 we specify A in detail.

For our proposed curve approximation scheme we need besides the rate between two control points also the distortion of every curve segment. Because every curve segment Q_k is defined by three control points, we let a general distortion measure for Q_k be d$d(p_{k-1}, p_k, p_{k+1})$. One popular distortion measure for curve approximations is the maximum absolute distance, which has also been employed in [5, 6, 2, 13]. Besides its perceptual relevance, this distortion measure has the advantage that it can be computed efficiently. Other distortion measures are evaluating the difference area between the original and the approximated boundary shape [14].

So far we have only discussed the edge distortion measures, i.e., the measures which judge the approximation of a certain partial boundary by a given curve segment. In general we are interested in a polygon distortion measure which can be used to determine the quality of approximation of an entire curve. We are using the maximum operator for the curve distortion, which is of the following form,

$$D(p_0, \ldots, p_{N_P+1}) = \max_{k \in [1, \ldots, N_P]} d(p_{k-1}, p_k, p_{k+1}), \tag{2}$$

Wheras the maximum absolute distance between a boundary and an edge of a polygon approximation can be exactly calculated with the formula for the distance between a point and a straight line, there is no unique method to measure the maximum distortion for a curve approximation of a boundary. However, the maximum distortion can be defined as the distance of every boundary point to its closest point of its approximated representation. If we imagine a "distortion-band" with the width $2 \cdot W_{max}$ along the boundary B a B-spline approximation must always be inside the band in order to full-fill the distortion requirement:

$$d(p_{k-1}, p_k, p_{k+1}) = \begin{cases} 0: & \text{all points of } Q_k(p_{k-1}, p_k, p_k) \text{ are inside distortion band} \\ D_{max}: & \text{any point of } Q_k(p_{k-1}, p_k, p_k) \text{ is outside distortion band} \end{cases}. \tag{3}$$

Fig. 1 shows how the distortion measure $d(p_{k-1}, p_k, p_{k+1})$ takes a curve segment Q_k given by the three control points $d(p_{k-1}, p_k, p_{k+1})$ as input and checks if all discrete curve points are within the distortion band of width $2 \cdot D_{max}$. The values of the curve points for this evaluation are not quantized to pixel resolution.

The following method describes a simple implementation of the distortion band. The distortion band is constructed by many overlapping squares of size $2 \cdot D_{max}$ along the boundary B. The distance between the squares must be below pixel resolution. The drawback of this method is that the thickness of the distortion band differs for a vertical or horizontal line and diagonal line. This shortcoming could be overcome by drawing circles instead of squares along B, but is computationally much more time intense.

In the reminder of the paper we introduce a fast and efficient algorithm which solves the following constrained optimization problem,

$$\begin{array}{l} \min_{p_0, \ldots, p_{N_P+1}} R(p_0, \ldots, p_{N_P+1}), \\ \text{subject to:} \quad D(p_0, \ldots, p_{N_P+1}) \leq D_{max}, \end{array} \tag{4}$$

where D_{max} is the maximum distortion permitted. Note that there is an inherent tradeoff between the rate and the distortion in the sense that a small distortion requires a high rate, whereas a small rate results in a high distortion. To find a B-spline curve approximation with the lowest possible distortion given a maximum rate R_{max},

$$\begin{array}{l} \min_{p_0, \ldots, p_{N_P-1}} D(p_0, \ldots, p_{N_P-1}), \\ \text{subject to:} \quad R(p_0, \ldots, p_{N_P-1}) \leq R_{max}, \end{array} \tag{5}$$

we use Eq.(4) iteratively.

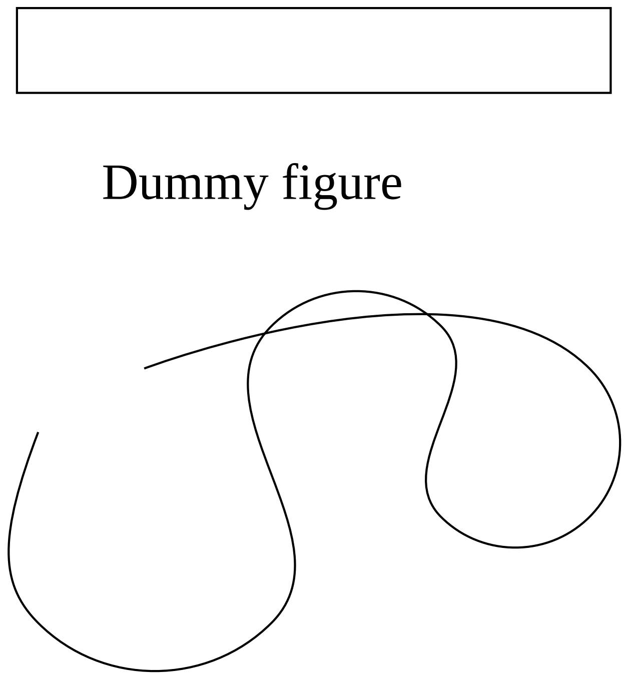

Figure 1: The distortion band is made out of a number of squares.

3 Review of B-spline curves

A B-spline is a specific type of curve from the family of parametric curves. A parametric curve with degree n consists of at least one curve segments where each curve segment is defined by $(n+1)$ control points. The control points are located around the curve segment and define its shape. A curve segment Q_k is defined in the form

$$Q_k(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix} \text{ with } 0 \le t \le 1, \tag{6}$$

The beginning and the end of a curve segment are called knots, they can be found with $t = 0$ respectively $t = 1$. A second degree curve segment $Q_i(t)$ is calculated as follows :

$$\begin{aligned} &Q_i(t) = T \cdot M \cdot P \\ &= \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \\ &\cdot \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{bmatrix} \cdot \begin{bmatrix} p_{k-1,x} & p_{k-1,y} \\ p_{k,x} & p_{k,y} \\ p_{k+1,x} & p_{k+1,y} \end{bmatrix} \end{aligned} \tag{7}$$

Both base matrix M, with specific constant parameters for each specific type of parametric curves, and control point matrix P with $(n+1)$ control points define the shape of Q_k in a two dimensional plane. Every point on the curve segment can now be calculated by letting t go from 0 to 1. The matrix product $T \cdot M$ is called the blending function B. The blending function determines how the three control points are weighted as a function of t. Figure 3 shows the blending function for a second degree B-spline curve. Among common parametric curves are the Bezier curve and the B-spline curve. For our boundary approximation algorithm we chose the second order (quadratic) basis uniform nonrational B-spline curve [15], Figure 2 a shows such a second order B-spline. We chose a B-spline with the lowest possible order to keep the the complexity of the curve to a minimum. A first degree B-spline is actually a polygon.

The easiest way to calculate a curve segment with a discrete number of curve points is to extend Eq.(7). If we let s be the number of discrete points of a curve segment we need s values between 0 and 1 for t. We redefine t, so that t is now a vector with s discrete values between 0 and 1 with the step size $\Delta t = \frac{1}{s-1}$: $t = \begin{bmatrix} 0 & \Delta t & 2\Delta t & \cdots & (s-1)\Delta t \end{bmatrix}'$. Applying vector t in Eq.(7) changes matrix T to a new size of 3 by s elements and matrix Q_k to 2 by s elements. Besides the just described matrix method B-splines can also be obtained in a recursive manner [16].

A second order $(n = 2)$ B-spline curve has the following properties:

- m: Total number of curve segments
- $(m+2)$: Total number of control points
- $(n-1) = 3$: control points per curve segment $Q_k : \{p_{k-1}, p_k, p_{k+1}\}$. p_{k-1} and p_k are also used by the previous curve segment Q_{k-1} and p_k and p_{k+1} as well by the next curve segment Q_{k+1}.
- $M = \begin{bmatrix} 0.5 & -1 & 0.5 \\ -1 & 1 & 0 \\ 0.5 & 0.5 & 0 \end{bmatrix}$: Base matrix M

Eq. (7) can now be represented by a function c that calculates a B-spline curve segment with three control points as input, and the matrix Q_k with s discrete curve points as output,

$$Q_k = c(p_{k-1}, p_k, p_{k+1}). \tag{8}$$

The beginning and at the end of the boundary approximation with a B-spline are special cases, because the first curve segment should start exactly from the first control point and the last curve segment should end exactly at the last control point. Using a double point $(p_{k-1} = p_k)$ the curve seg-

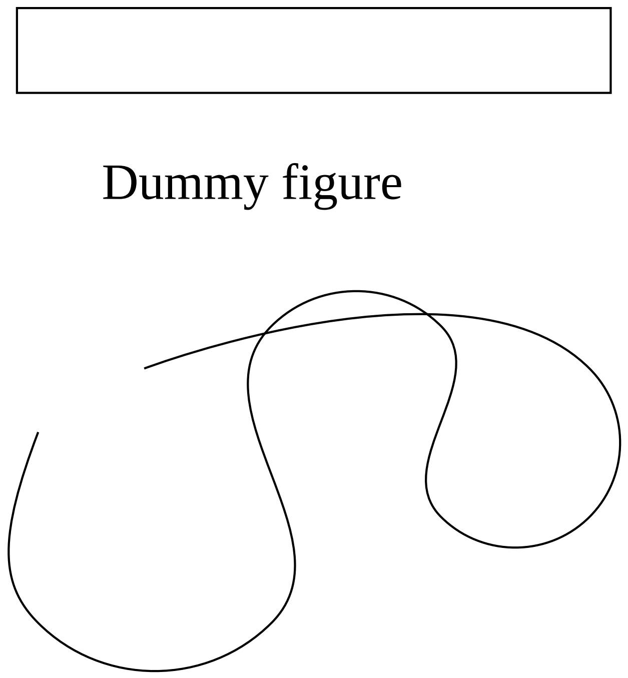

Figure 2: A second degree B-spline curve. The first two control points are exactly on the same location (a double point), so that the first curve segment starts from that double point.

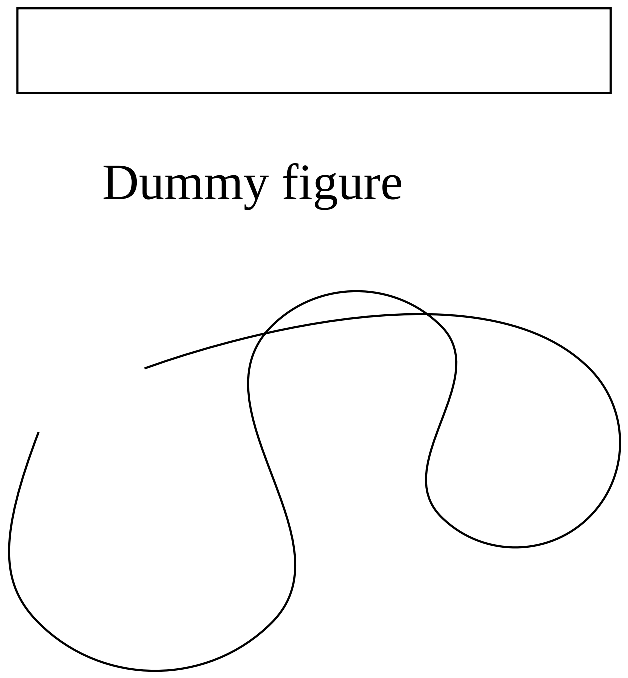

Figure 3: Blending function of a second degree B-spline.

ment Q_k will begin exactly from the double point (Figure 2). We apply this property at the beginning and at the end of the curve, so that $p_0 = p_1$ and $p_{N_P} = p_{N_P+1}$. These two special cases can easily be incorporated into the boundary approximation algorithm.

4 Admissible control point set

From a theoretical point of view, the set of admissible control points for a B-spline boundary approximation should contain all the pixels in the image plane. In order to keep the algorithm efficient we restrict the control points to a set of relevant locations. We call this set of the admissible control points A and define it as a band along the boundary B with the width $2 \cdot W_{max}$ (Figure 4). Set A must be an ordered set to employ the proposed algorithm. We propose therefore to order set A by assigning all points of A to their nearest boundary point. Every admissible control point a_{i,i_b} has two indexes. The first index i has the same number as the index of its closest boundary point b_i. The second index i_b numerates the admissible control points with the same index i. Index i_b starts always with 0, every point $a_{i,0}$ with $i_b = 0$ is by definition b_i, therefore so if $W_{max} = 0$ then $A = B$. Clearly, not every boundary point has the same number of admissible control points. Good values for W_{max} are 1.0, this results in a band with a thickness of 3 pixels. W_{max} is measured from the center of the boundary pixel to the center of the admissible control point pixel. We further define that the first and the last boundary points have no additional admissible control points assigned.

5 The minimum rate case

The goal of the proposed algorithm is to find the B-spline curve whose control points can be encoded with the smallest number of bits under two conditions: 1) the distortion of the curve is smaller or equal to the maximum distortion D_{max} as stated in Eq. (4) and 2) the control points must be selected from the admissible control point set A.

The key observation for deriving an efficient search is the fact that given a certain control point (p_k) of a B-spline curve and the rate which is required to code the curve up to and including this control point $R_k(p_k)$, the selection of the next control point p_{k+1} is independent of the selection of the previous control points $p_0, \ldots, p_k$. This is

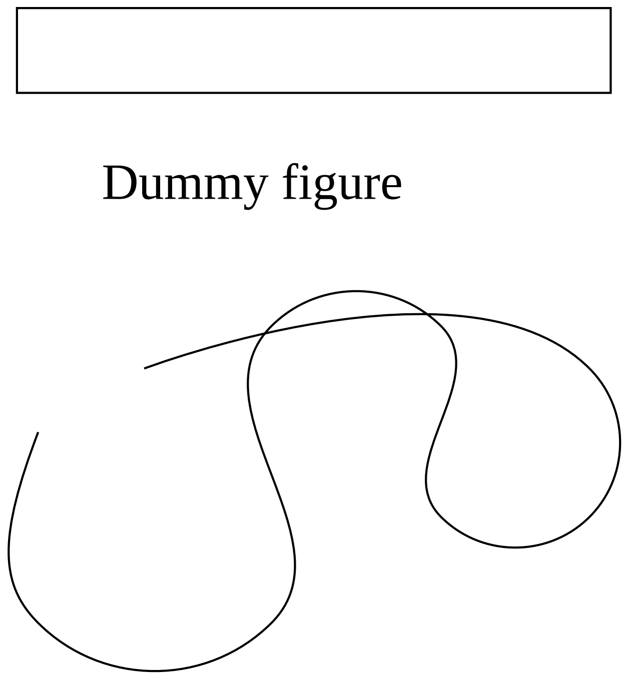

Figure 4: The admissible control point set A. Sample set.

true since the total rate R can be expressed recursively as a function of the control point rates $r(p_k, p_{k+1})$. That is,

$$R_{k+1}(p_{k+1}) = R_k(p_k) + r(p_k, p_{k+1}), \qquad (9)$$

A specific encoding scheme $r(p_k, p_{k+1})$ is discussed in the next section. However, the distortion of the new curve segment Q_k depends not only on the current control point p_k and the next control point p_{k+1} but also on the previous control point p_{k-1} (Figure 6). The weight function w combines Eq. (9) and the distortion measure d Eq. (3), where

$$w(p_{k-1}, p_k, p_{k+1}) = \begin{cases} \infty : \\ d(p_{k-1}, p_k, p_{k+1}) > D_{max} \\ \\ r(p_k, p_{k+1}) : \\ d(p_{k-1}, p_k, p_{k+1}) \leq D_{max} \end{cases} . \qquad (10)$$

The recursion of Eq. (9) needs to be initialized by setting $R_{-1}(p_{-1})$ to zero. Note that by the definition of w, the rate for a curve segment which does not satisfy the maximum distortion constraint is infinite. Clearly $R_{N_P+1}(p_{N_P+1}) = R(p_0, \ldots, p_{N_P+1})$, is the rate for the entire curve.

As indicated above, we need to start the search for an optimal polygon at a given vertex. If the boundary is not closed, the first boundary point b_0 has to be selected as the first vertex p_0. For a closed boundary, we can select any boundary point. Note that our experiments have shown that the efficiency of the solution approach does not depend on to the choice of the initial starting point.

The problem in Eq. (4) can be formulated as a shortest path problem in a weighted directed graph, (Fig. **??**). A vector $\vec{E}$ starts at control point $p_u = a_{i,i_b} \in A$ and ends at control point $p_v = a_{k,k_b} \in A$ with the condition that both control points cannot be assigned to the same boundary point $\{\vec{E}(a_{i,i_b}, a_{k,k_b}); \forall i \neq j\}$ (see Fig. **??**). A path of order K from control point p_0 to a control point p_K is an ordered set $\{p_0, \ldots, p_K\}$. The length of a path is defined as follows,

$$\sum_{k=1}^{K-1} w(v_{p-1}, p_k, p_{k+1}), \qquad (11)$$

Again, note the above definition of the weight function leads to a length of infinity for every path which includes a curve segment resulting in an approximation error larger than D_{max}. Therefore a shortest path algorithm will not select these paths.

The classical algorithm for solving such a single-source shortest-path problem, where all the weights are non-negative, is the Dijkstra's algorithm [17]. This is a significant reduction compared to the time complexity of the exhaustive search. We can further simplify the algorithm by observing that it is very unlikely for the optimal path to select a control point $p_u = a_{k,k_b}$ when the last selected control point was $p_v = a_{i,i_b}$, where $i > k$. Hence the vector set ε is redefined in the following way, $\{\vec{E}(a_{i,i_b}, a_{k,k_b}); \forall i < k\}$ (see Fig. **??**). This restriction results in that the selected curve approximation has to follow the original boundary without rapid direction changes, and more important, the resulting graph is a weighted directed acyclic graph (DAG). For a DAG, the DAG-shortest-path algorithm [17] finds a single-source shortest-path and is even faster than Dijkstra's algorithm.

Let $R^*(\{a_{j,j_b}, a_{i,i_b}\})$ be a four dimensional array that represents the minimum total rate to reach the control point $p_u = a_{i,i_b}$ from the source control point $p_0 = a_{0,0}$ via a B-spline curve approximation. $p_u = a_{i,i_b}$ is the current control point and $p_{u-1} = a_{j,j_b}$ is the previous control point of a curve segment. Lets further define control point p_u as a state, and $\{p_{u-1}, p_u\}$ as p_u's state value. To calculate the rate r between a_{i,i_b} and a future control point $p_{u+1} = a_{k,k_b}$ we don't need to know the location of the previous control point a_{j,j_b}. We need a third point a_{j,j_b} to calculate the weight w with Eq. (10), so in R^* there is for every state value the lowest total rate. Clearly the lowest state value $R^*(\{a_{j,j_b}, a_{N_B-1,0}\})$ of state $p_{N_P+1} = a_{N_B-1,0}$ is the solution to problem (4). Function $c(x)$ on lines (5),(8) and (11) returns the number of admissible control points that are assigned to the boundary point b_x. Let $q()$ be a back pointer which is used to remember the optimal path, $q()$ points from one state value to another. Then the proposed algorithm works as follows:

```
    DAG shortest path algorithm
1)  assign all $R^*(\{a_{j,j_b}, a_{i,i_b}\})$ to $\infty$;
2)  $R^*(\{a_{0,0}, a_{0,0}\}) = 0$;
3)
4)  for $i = 0, \dots, N_B - 2$;
5)  { for $i_b = 0, \dots, c(i)$;
6)    {
7)      for $j = i - l_{sw}, \dots, i - 1$;
8)      { for $j_b = 0, \dots, c(j)$;
9)        {
10)         for $k = i + 1, \dots, i + l_{sw}$;
11)         { for $k_b = 0, \dots, c(k)$;
12)           {
13)             $p_{u-1}$ $=a_{j,j_b}$;
14)             $p_u$ $=a_{i,i_b}$;
15)             $p_{u+1}$ $=a_{k,k_b}$;
16)             distortion of $Q_u$: $d(p_{u-1}, p_u, p_{u+1})$;
17)             rate: $r(p_u, p_{u+1})$;
18)             weigth: $w_u = w(p_{u-1}, p_u, p_{u+1} b_i, b_j)$;
19)             if $R^*(\{p_{u-1}, p_u\} + w_u$
20)             $< R^*(\{p_u, p_{u+1}\})$;
21)             {
22)               store new lowest rate:
23)               $R^*(\{p_u, p_{u+1}\}) =$
24)               $R^*(\{p_{u-1}, p_u\}) + w_u$;
25)               assign back pointer to previous
26)               state value:
27)               $q(\{p_u, p_{u+1}\}) = \{p_{u-1}, p_u\}$;
28)             }
29)         } }
30)     } }
31) } }
32) find the state value in the last state
33) with the lowest rate:
34) $R_{winner} = \infty$;
35) for $j = i - l_{sw}, \dots, i - 1$;
36) { for $j_b = 0, \dots, c(j)$;
37)   {
38)   if $R^*(\{a_{j,j_b}, a_{N_b,0}\}) < R_{winner}$;
39)       $q_{winner} = \{a_{j,j_b}, a_{N_b,0}\}$;
40)   }
41) }
```

The optimal path $\{p_0^*, \dots, p_{N_P+1}^*\}$ can be found by back tracking the pointers $q()$ in the following recursive fashion (by definition $\{p_{N_P}^*, p_{N_P+1}^*\} = q_{winner}$),

$$\{p_{k-1}, p_k\}^* = q(\{p_k, p_{k+1}\}), k = N_P - 1, \dots, 2. \quad (12)$$

The formal proof of the correctness of the DAG-shortest-path algorithm, on which the above scheme is based, can be found in [17]. We will reason more intuitively how this approach works. In line (1) the minimum rate to reach any of the control point R^* is set to infinity for all state values, except all R^* of the state values of state $a_{0,0}$ are set to zero.

The two "for loop"s (counter variable i and i_b) in line (4) and (5) select the all control points in A in sequence as p_u of curve segment Q_u and the two "for loop"s in line (10) and (11) (counter variable k and k_b) select all possible control points for p_{u+1}. Hence these four "for loops" select each possible vector $\vec{E}$ in the DAG exactly once. The double loop on line (7) and (8) (counter variable j and j_b) is necessary to measure the distortion of Q_u. Since the weigth function w needs three control points, three double loops are necessary to cover every possible curve segment in the DAG. A DAG shortest path algorithm for a polygon needs only two double loops because an edge segment is defined by two vertex points. The reason for the "double loops" is that we need two loops in order to cover all admissible control points in A. The number of loops of the "for loop"s on line (7) and (10) depend on the length l_{sw} of the sliding window. Since the lines (12) to (28) are processed for every curve segment. The lines (16) to (18) are used to calculate the weight of the edge, $w(p_{u-1}, p_u, p_{u+1})$. The most important part of this algorithm is the comparison on line (19). Here we test if the new bit rate, $R^*(p_u) + w(p_{u-1}, p_u, p_{u+1})$, to reach the admissible control point p_{u+1}, given that the last

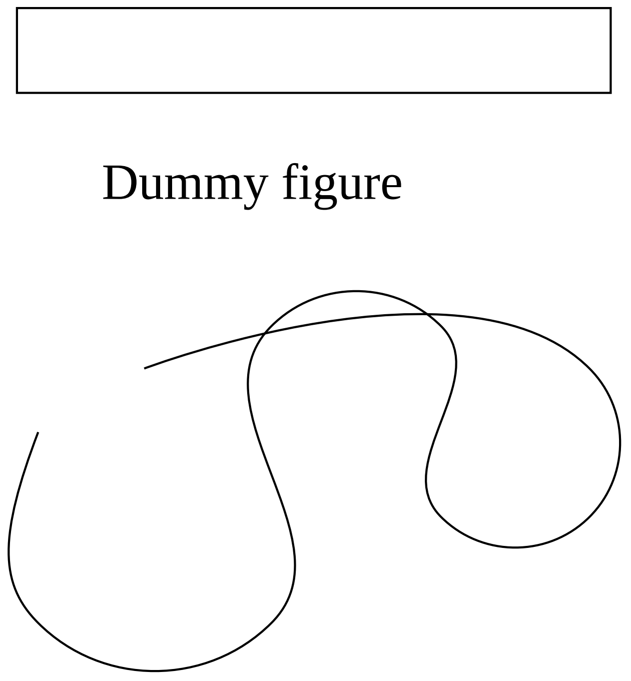

Figure 5: A B-spline curve segment.

two control points were p_{u-1} and p_u, is smaller than the smallest bit rate used so far to reach the control point p_{u+1}, $R^*(\{p_u, p_{u+1}\})$. If this bit rate is indeed smaller, then it is assigned as the new smallest bit rate to reach the control point p_{u+1}, $R^*(\{p_u, p_{u+1}\}) = R^*(\{p_{u-1}, p_u\}) + w_u$ on line (23). We also assign the back pointer of state value $\{p_u, p_{u+1}\}$ to state value $\{p_{u-1}, p_u\}$. This algorithm leads to the optimal solution because, as stated earlier, when the rate $R^*(\{p_{u-1}, p_u\})$ of two control points $p_{u-1} = a_{j,j_b}$ and $p_u = a_{i,i_b}$ are given, then the selection of the future control points ($p_{u+1} = a_{k,k_b}$; $i < k < N_B$) is independent of the selection of the past control points ($p_x = a_{x,x_b}$; $0 \leq x < i$).

6 The minimum distortion case

We now consider the minimum distortion case which is stated in Eq. (5). We propose an iterative solution to this problem which is based on the fact that we can solve the dual problem stated in Eq. (4) optimally. Consider D_{max} in Eq. (4) to be a variable. We derived in the previous section an algorithm which finds the polygonal approximation which results in the minimum rate for any D_{max}. We denote this optimal rate by $R^*(D_{max})$. $R^*(D_{max})$ is a non-increasing

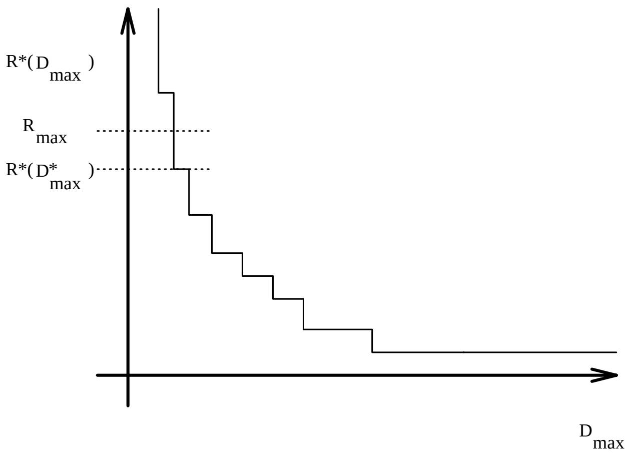

Figure 6: The $R^*(D_{max})$ function, which is a non-increasing function exhibiting a staircase characteristic. The selected R_{max} falls onto a discontinuity and therefore the optimal solution is of the form $R^*(D^*_{max}) < R_{max}$, instead of $R^*(D^*_{max}) = R_{max}$.

function (for a prove see [18]), we can use bisection [19] to find the optimal D^*_{max} such that $R^*(D^*_{max}) = R_{max}$. Since this is a discrete optimization problem, the function $R^*(D_{max})$ is not continuous and exhibits a staircase characteristic (see Fig. **??**). This implies that there might not exist a D^*_{max} such that $R^*(D^*_{max}) = R_{max}$. In that case the proposed algorithm will still find the optimal solution, which is of the form $R^*(D^*_{max}) < R_{max}$, but only after an infinite number of iterations. Therefore if we have not found a D_{max} such that $R^*(D_{max}) = R_{max}$ after a given maximum number of iterations, we terminate the algorithm.

7 Control point encoding scheme

So far we have not assumed any specific scheme for encoding the vector between two control points of the curve. Function $r(p_u, p_v)$ needs a specific control point encoding scheme to compute the rate between two control points. In this section we present a control point encoding scheme which can be considered a combination of an modified 8-connect chain code and a run-length encoding scheme. The chain code and the run-length encoding can be combined by representing the vector between two control points by an angle α and a run β, which form the symbol

run	Codeword	run	Codeword
1	00	8	11000
2	010	9	11001
3	011	10	11010
4	1000	11	11011
5	1001	12	11100
6	1010	13	11101
7	1011	14	11110
		15	11111

Table 1: Code word assignment for the runs β (length of control point vector)

(α,β). For each of the possible symbols (α,β) we encode the angle and the run independently. In this paper, we employ a logarithmic code for encoding the runs (βs), which is displayed in Table 1. Note that the longest possible run is 15.

Clearly it takes 3 bits to encode 8 equally probable directions of α. α is an angle of 45 degree increments, therefore there are only 8 possible values for α. In natural boundaries, the arrival direction of the vector is highly correlated with the departure direction of the following vector. This implies that the arrival direction should be used to predict the departure direction. We propose to use only 2 bits for α for the 4 most probable directions. The four directions of α are $+45^o, +90^o, -45^o, -90^o$, where 0^o is the direction of the previous vector. α is not anymore an absolute angle but a relative angle one. Even if we are using the 8-connect chain code for encoding arbritray vectors, all the vectors with angles that are not multiples of 45 degree cannot be encoded. With the new scheme to encode angles we restrict the set of codeable vectors slightly, but on the other hand gain lower encoding rates for the control point vectors.

The curve segment rate function $r()$ must also consider the case when a vector cannot be encoded. If this happens, the output is infinity,

$$r(p_u, p_v) = \begin{cases} \text{length of } (\alpha + \beta) : \vec{E} \text{ is codeable} \\ \infty : \vec{E} \text{ is not codeable} \end{cases}$$

$$\alpha : 2 \text{ bits}, \beta : 2 \text{ to } 5 \text{ bits} \tag{13}$$

Experiments showed that using an encoding scheme with 4 relative angles results in the boundary approximation with lower rates than when we used the scheme with 8 absolute angles. The better results were achieved with both the polygon and the B-spline approximation.

8 Experimental results

9 Summary

References

[1] H. Musmann, M. Hötter, and J. Ostermann, "Object-oriented analysis-synthesis coding of moving images," *Signal Processing: Image Communication*, vol. 1, pp. 117–138, Oct. 1989.

[2] M. Hötter, "Object-oriented analysis-synthesis coding based on moving two-dimensional objects," *Signal Processing: Image Communication*, vol. 2, pp. 409–428, Dec. 1990.

[3] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 260–268, June 1961.

[4] J. Saghri and H. Freeman, "Analysis of the precision of generalized chain codes for the representation of planar curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, pp. 533–539, Sept. 1981.

[5] J. Koplowitz, "On the performance of chain codes for quantization of line drawings," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, pp. 180–185, Mar. 1981.

[6] D. Neuhoff and K. Castor, "A rate and distortion analysis of chain codes for line drawings," *IEEE Transactions on Information Theory*, vol. IT-31, pp. 53–68, Jan. 1985.

[7] T. Kaneko and M. Okudaira, "Encoding of arbitary curves based on the chain code representation," *IEEE Transactions on Communications*, vol. COM-33, pp. 697–707, July 1985.

[8] R. Prasad, J. W. Vieveen, J. H. Bons, and J. C. Arnbak, "Relative vector probabilities in differential chain coded line-drawings," in *Proc. IEEE Pacific Rim Conference on Communication, Computers and Signal Pro-*

cessing, (Victoria, Canada), pp. 138–142, June 1989.

[9] A. K. Jain, *Fundamentals of digital image processing.* Prentice-Hall, 1989.

[10] E. J. R. Myron Flickner, James Hafner and J. L. Sanz, "Periodic quasi-orthogonal spline bases and applications to least-squares curve fitting of digital images," *ip*, vol. 5, pp. 71–88, Jan. 1996.

[11] C. P. Quist, "Intra-frame contour compression with cubic b-splines," Master's thesis, Delft University of Technology, 1995.

[12] T. Özçelik, *A Very Low Bit Rate Video Codec.* PhD thesis, Dept. EECS, Northwestern University, Dec. 1994.

[13] G. M. Schuster and A. K. Katsaggelos, "An optimal lossy segmentation encoding scheme," in *Proceedings of the Conference on Visual Communications and Image Processing*, pp. 1050–1061, SPIE, Mar. 1996.

[14] tech. rep., MPEG 4 Shape coding ad-hock group, 1996.

[15] Foley, vanDam, Feiner, and Hughes, *Computer Graphics: Principles and Practice*, pp. 478–516. Addison-Wesley, 1990.

[16] F. S. Hill, *Computer graphics.* Macmillan Publishing Company, 1990.

[17] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to algorithms.* McGraw-Hill Book Company, 1991.

[18] G. M. Schuster, *A video compression scheme with optimal bit allocation among segmentation, motion and residual error.* PhD thesis, Northwestern University, Evanston, Illinois, USA, June 1996. Department of Electrical Engineering and Computer Science.

[19] C. F. Gerald and P. O. Wheatley, *Applied numerical analysis.* Addison Wesley, fourth ed., 1990.