Appl. Math. and Comp. Sci., 1993, vol.3, No.2, 301–311

DATA REDUCTION FOR THE REPRESENTATION OF CURVES BY PIECEWISE POLYNOMIALS

Christoph Maas*, Frank Levermann**

In the course of the development of an industrial CAD–software package a method had to be developed for storing curves given by a scanner in bitmap format. The objective is to reduce the amount of data necessary for defining the curve while allowing a certain deviation of the stored curve from the original one. Two solutions are presented based upon Hermite–interpolation or B–Spline-interpolation of the bitmap, respectively. The solutions involve questions about an appropriate parametrization of the data and about easy editing of the results by the user of the program.

1. Introduction

In the course of the development of an industrial CAD–software package a method had to be developed for storing data resulting from the inspection of curves by a scanner. For a given discretization of the X–Y–plane a curve is represented by a bitmap, i.e. a sequence of (x, y)–coordinates of those pixels recognized by the scanner as lying on the curve. To facilitate further handling of the curve this representation should be converted into vector format, which at the same time reduces the number of data needed and gives better control over the shape (especially over the smoothness) of the curve.

A class of functions frequently used for this purpose are piecewise polynomials. In this paper it is the objective to find a practically useful representation of the curve needing as few data as possible. Since the scanning process necessarily involves a certain amount of imprecision, this representation is allowed to deviate slightly from the bitmap data. The feasible difference between the two curves is described in the form of constraints for every element of the bitmap.

Related problems have been considered in the literature by a certain number of authors. They usually give spline representations of the curve using B–splines as basis functions. While some papers (Kjellander, 1983a; 1983b; Farin *et al.*, 1987) deal with the removal of one knot, in (Lyche and Mørken, 1987; 1988) there is a comprehensive strategy for the removal of a large number of knots. A different representation of the curve based on cubic Hermite interpolation can be found in

* Fachhochschule Hamburg, Fachbereich Elektrotechnik und Informatik, Berliner Tor 3, 20099 Hamburg, Germany
** Kondata GmbH, Jessenstr. 3, 22767 Hamburg, Germany

(Wever, 1989), again dealing with the removal of as many knots as possible. Since the strategies for knot removal in (Lyche and Mørken, 1987; 1988) and in (Wever, 1989) both are of a heuristical nature, it needs to be studied experimentally how they behave in practical application situations.

The problem considered in this paper, however, is somewhat different from the standard situation handled by those authors. Since the input data are taken from a bitmap, the coordinates of two subsequent points differ by one unit from their $x-$ or $y-$coordinates. Additionally, the pair of coordinates of a certain pixel represents an original curve point lying anywhere in the area covered by the pixel. So there is no reason for allowing only very small deviations from the bitmap data during the approximation process. This means that compared with the example data from (Lyche and Mørken, 1987; 1988; Wever, 1989) we are dealing with points that are relatively far apart from each other and are allowed to be moved around by a relatively large amount. Therefore, it is even more worth investigating whether the algorithms found in the literature give useful results in this situation.

In the sequel of this paper, first the problem of the parametrization of the bitmap data is addressed. Then both data reduction strategies are introduced, and they are applied to a demonstration example. Finally, some questions concerning the practical application of the algorithms are discussed, and the B-spline representation is computed for a large real-world example.

2. Parametrization of the Data

The parametrization of the data has a large influence on the shape of the curve constructed from these data. There are several possibilities for the parametrization of curves given as sequences $(x_i, y_i)_{i=0}^n$ of pairs of $(x, y)-$values. Throughout this text t_i will always denote the parameter value assigned to x_i and y_i.

2.1. Uniform and Chord-Length Parametrizations

Widely used techniques for the parametrization of data are the uniform parametrization where the parameter only reflects the numbering of the points (i.e. $t_i := i$) and the chord-length parametrization defined by

$$t_i := \begin{cases} 0 & \text{for } i = 0 \\ t_{i-1} + \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} & \text{otherwise} \end{cases}$$

Both parametrization methods are not influenced by the shape of the curve.

2.2. Variable-Speed Parametrization

Several methods have been suggested in order to establish a closer connection between the behaviour of the curve in the neighbourhood of the point (x_i, y_i) and its parameter value at this point. By the strategy of Cohen and O'Dell (1989) the sequence $(x_i, y_i)_{i=0}^n$ is divided into segments each of which can be approximated by a straight line within a given tolerance, and the parametrization of the points of one segment is determined with respect to the length of its approximating line and the angle it forms with its predecessor and with its successor. As a result, the parameter values rise rather slowly in those parts of the curve where it resembles a straight line, while parts of the curve showing relatively large changes of direction have rapidly increasing parameter values (and thus, are passed in slow motion when traversing the curve).

For our purposes, this method turned out to be very useful, because it produces not only a parametrization of the given data, but also gives first information about the overall structure of the curve.

3. Representation by Cubic Hermite–Interpolation

3.1. Basic Concepts

One possibility for the representation of the curve given by the discretization $(x_i, y_i)_{i=0}^n$ would be to interpolate the points (x_i, y_i) and (x_{i+1}, y_{i+1}) by a pair of cubic polynomials

$$\begin{pmatrix} X_i(t) \\ Y_i(t) \end{pmatrix}, \quad t \in [t_i, t_{i+1}]$$

such that

$$\begin{pmatrix} X_i(t_i) \\ Y_i(t_i) \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \text{ and } \begin{pmatrix} X_i(t_{i+1}) \\ Y_i(t_{i+1}) \end{pmatrix} = \begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix}$$

and additionally the curve

$$\begin{pmatrix} X(t) \\ Y(t) \end{pmatrix}, \quad t \in [t_0, t_n]$$

defined by

$$\begin{pmatrix} X(t) \\ Y(t) \end{pmatrix} := \begin{pmatrix} X_i(t) \\ Y_i(t) \end{pmatrix}, \quad \text{if } t \in [t_i, t_{i+1}]$$

is C^1–continuous.

This last property can be achieved by choosing[1] values $\dot{x}_i$, $\dot{y}_i$ $(i = 0, \dots, n)$ and demanding that for all $i = 0, \dots, n-1$ we have

$$\begin{pmatrix} \dot{X}_i(t_i) \\ \dot{Y}_i(t_i) \end{pmatrix} = \begin{pmatrix} \dot{x}_i \\ \dot{y}_i \end{pmatrix} \text{ and } \begin{pmatrix} \dot{X}_i(t_{i+1}) \\ \dot{Y}_i(t_{i+1}) \end{pmatrix} = \begin{pmatrix} \dot{x}_{i+1} \\ \dot{y}_{i+1} \end{pmatrix}$$

[1] Though it is possible to choose the derivatives $\dot{x}_i$ and $\dot{y}_i$ arbitrarily, there are certain problems that should be considered in order to construct reasonable interpolation curves (see (Wever, 1989, pp. 68–71) for a survey of appropriate strategies for the choice of values for the derivatives).

3.2. Data Reduction

When such a representation has been found, it can then be asked which one of the pairs (x_i, y_i) can be removed from the sequence $(x_i, y_i)_{i=0}^n$ such that Hermite interpolation for the remaining points still gives an acceptable representation of the original curve.

An algorithm for this approach is given in (Wever, 1989):

For each of the values x_i and each of the values y_i the user has to specify by how much the $x-$ and the $y-$values of the simplified curve at $t = t_i$ are allowed to differ from the original data.

After having been ordered according to an estimate of how much the shape of the curve will be affected by leaving out this particular point, the points (x_i, y_i) are inspected for possible removal.

To this purpose, let (x_L, y_L) and (x_R, y_R) be two points that have not yet been removed with L being the highest number less than i and R the lowest number greater than i.

The point (x_i, y_i) will be removed from the sequence of points, if it is possible to approximate (x_L, y_L) and (x_R, y_R) by a pair of cubic polynomials

$$\begin{pmatrix} X_L(t) \\ Y_L(t) \end{pmatrix}, \ t \in [t_L, t_R]$$

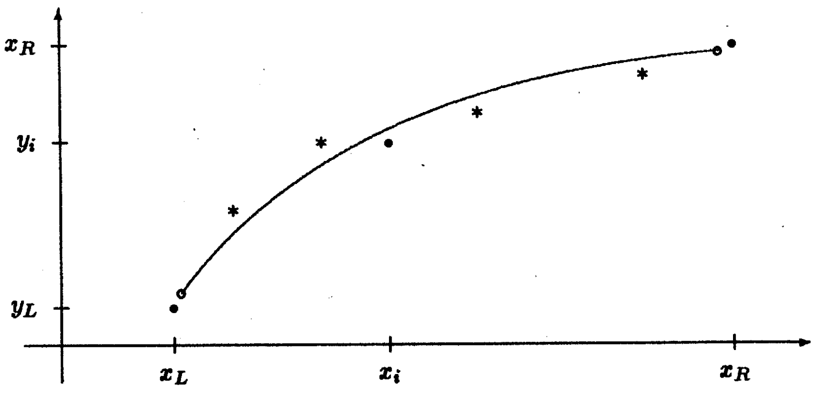
such that all the values $X_L(t_j) - x_j$ and $Y_L(t_j) - y_j$ for $j = L, \ldots, R$ are within the tolerances (see Figure 1), and C^1-continuity is preserved for the interfaces to the neighbouring polynomials.

3.3. An Example

The outline of the letter "B" is described by a sequence of 29 points in the plane (marked as "×" in Figure 2 – it should be noticed that several points appear twice in this sequence; the first and last point is in the upper left corner). The horizontal (or vertical) distance between two neighbouring pixels is taken as unity. Using the chord–length parametrization and applying the data reduction algorithm with a maximal deviation from the original point of 0.1 allows a representation of the curve with only 10 points, while for a tolerance of 0.5 it suffices to construct the approximating curve from 5 points.

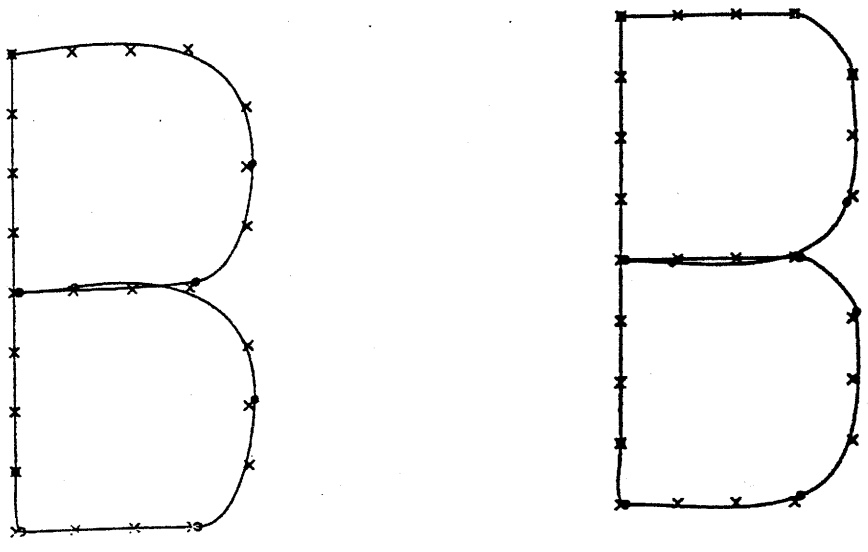
Using the variable–speed parametrization gives slightly different results. A tolerance of 0.1 allows a reduction of the number of points to 12, while a tolerance of 0.5 reduces the number of required points to 6.

For both parametrizations, only the computation with a low tolerance gives a useful representation of the original curve (and is therefore shown in Figure 2), and in this case the result produced by using the variable speed parametrization is more faithful to the original shape of the curve.



● = original data
o = data changed during removal process
∗ = points already removed

Fig. 1. Testing (x_i, y_i) for possible removal during Hermite interpolation.



chord–length parametrization variable speed parametrization

× : original data
o : endpoints of the cubic polynomials after the data reduction

Fig. 2. Data Reduction applied to an Hermite–interpolation of the given discretization of the letter "B".

4. B–Spline Representations

The authors of (Lyche and Mørken, 1987; 1988) give a strategy for transforming an interpolating B–spline[2] into an approximating B–spline with a lower number of knots and control points while observing a given maximal distance between the approximating and the interpolating spline (so as soon as the spline interpolation has been set up, all approximation is measured with respect to the complete spline and not only the points from the original discretization). For the present purpose this strategy could be employed in the following way:

4.1. Linear Approximation

In the first step, the bitmap data are interpolated by a B–spline of order $k = 2$, i.e. a polygon. Then the data reduction algorithm is applied tolerating a rather high deviation of the resulting curve from the original (approximately one–half of the diagonal distance of two pixels). This removes a great part of the tension built up by the need to round the real world coordinates of the points of the curve to pixel coordinates. The resulting list of knots and control points is the basis for the procedure in the next paragraph.

4.2. Cubic Approximation

Secondly, by taking every internal knot three times and introducing two new control points on each line connecting two successive control points the C^0–curve obtained in the last paragraph can now be written as a cubic B–spline (i.e. order $k = 4$) (Lyche and Mørken, 1988, p. 200). Another application of the data reduction algorithm produces an approximation to the curve obtained so far which first of all has fewer knots and control points (thus compensating for the introduction of new data at the beginning of this step). Additionally, however, in this new curve there may occur C^1– or even C^2–continuity at the meeting point of two polynomials, since the data reduction may also reduce the multiplicity of the knots hereby raising the order of continuity.

The tolerance used here was approximately one–half of the horizontal distance between two neighbouring pixels. A significantly lower tolerance forces the new approximation to follow every "whim" of the curve obtained so far and therefore often produces a spline with frequent oscillations, while allowing larger deviations between the curves did not seem to have much influence on the result at this stage (see also the differences between the two results in Figure 3).

Applying the reduction procedure in (Lyche and Mørken, 1988) to closed curves requires quite a few modifications but does not face any serious obstacles.

[2] For general information about B–Splines see (de Boor, 1978, pp. 108 ff.) or (Hoschek and Lasser, 1989, pp. 157 ff.).

4.3. "Windfall Profits" from the Parametrization Strategy

When carrying out this data reduction strategy the parametrization procedure (Cohen and O'Dell, 1989) turned out to be particularly useful in two ways.

First, this parametrization gives hints about where to localize corners of the given curve. If any reduction of the multiplicity of the corresponding entries in the knot vector is being forbidden, the resulting curve will remain only C^0–continuous at these points.

Second, the parametrization suggests a splitting up of the original curve into several segments which can then be handled independently by the data reduction algorithm. Since in each internal step of the reduction algorithm it is tried to select data evenly distributed over the whole curve (cf. Lyche and Mørken, 1988, p. 192), the complete set of data has to be accessible. Depending on the memory size of the computer available this might result in a certain amount of paging between the main memory and other storage devices. So, this method can become rather slow for long curves and does therefore experience a considerable speed–up when being applied to each segment of the curve independently.

4.4. Example (continued)

In order to allow a comparison between the two strategies the B–spline based reduction technique was applied to the example curve of section 3, too. Because the original curve is represented by an untypically low number of points, the tolerance in the first (i.e. linear approximation) step had to be lower than in a general case. As before, the data reduction was performed with a low and a large tolerance, alternatively. Allowing a tolerance of zero in the linear approximation step and a tolerance of 0.1 in the cubic approximation step results in a representation having 29 knots (since there are several multiple knots, the resulting curve indeed consists of 16 polynomials). Enlarging the tolerances to 0.2 in the first and 0.5 in the second step results in a B-spline representation with 17 knots (being composed of 6 polynomials). Here the larger tolerance values, too, lead to useful approximation curves (see Figure 3), so the B–spline approach results in a representation of the given curve using considerably less data than the Hermite polynomial approach.

5. Final Remarks

5.1. Pre-Processing of the Input Data

5.1.1. Interpretation of the Pixel Coordinates

Every pair of (x, y)–coordinates of the bitmap actually stands for one pixel, i.e. a (small) surface the area of which is determined by the given discretization of the plane. There are two different ways for the interpretation of bitmap data (see Figure 4): Every pixel given in the bitmap can either be considered as a representative for its center point or as a representative for a curve covering some part of the circumference of the pixel.

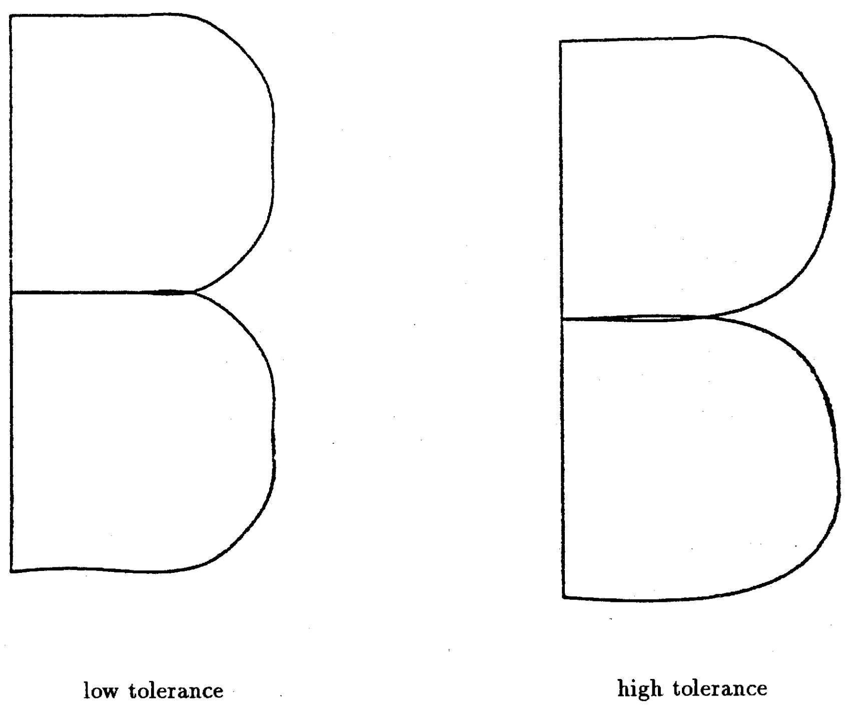


Fig. 3. The same example as in the last section, now represented by B– Splines (using variable speed parametrization).

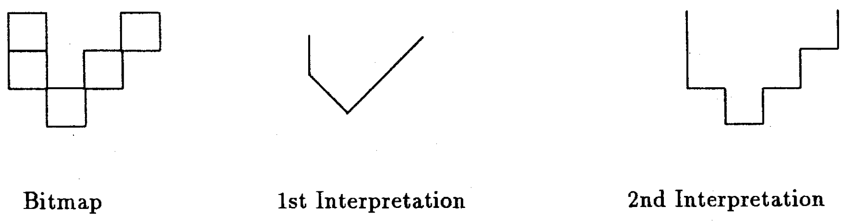


Fig. 4. Two ways of interpreting the pixels in a bitmap.

According to our experience, the first interpretation tends to give better results, since the staircase shape obtained by the second interpretation requires a lot of smoothing effort to give a convincing curve.

5.1.2. Scanning Unprecisely Drawn Lines

We assume that the drawing handed over to the scanner is a clear black and white picture. So we have not implemented any techniques for handling fuzzy outlines. In certain cases, this may of course lead to an improper representation of the curve. But since we preferred to concentrate all opportunities for the user to influence the whole process manually at one point, we decided not to introduce any outside manipulation of the data at this stage.

5.2. Comparison of Hermite and B–Spline Approximations

The example that was used to demonstrate the results of the different techniques for data reduction also clearly shows the main differences between the two approaches: Data reduction applied to an Hermite interpolation always forces the result to be C^1–continuous, while using B–splines yields a curve that is C^2–continuous in general, but can be forced to have only C^1– or C^0–continuity at particular points. Furthermore, the B–spline approach gives useful curves even for higher tolerances thus allowing a more efficient data reduction.

5.3. A Real–World Example

The picture in Figure 5 was scanned with a resolution of 144 dpi × 144 dpi. The procedure for setting up the bitmap reported 569 objects the outlines of which are described by 60766 pairs of (x, y)–values. Using the B–Spline representation, the reduction algorithm was run with a tolerance of 0.71 for the linear approximation and 0.5 for the cubic approximation. The result was a description of the curves using 11667 pairs of (x, y)–values (cf. Figure 6) and consisting of 4983 polynomials. The data reduction needed 382 seconds of CPU time on a Vax Station 4000 VLC with 24 MB RAM.

5.4. Post–Processing of the Result by the User

Practically every picture with a certain amount of complexity contains points for which the question which curve they belong to can only be answered when you understand the semantics of the picture. So we have to accept that any automatic processing is very likely to produce an error here. Therefore, after the data reduction has been carried out the user is given the opportunity to edit the resulting curve. It is possible to transform the curve into any format required by the user, e.g. we implemented the transformation into the IKARUS format (Karow, 1986), which is rather popular in the area of digitally describing character fonts and allows easy editing. Even when taking into account the amount of time necessary for this extra work, the present algorithm means a considerable saving of time compared with manually digitizing a complete picture.

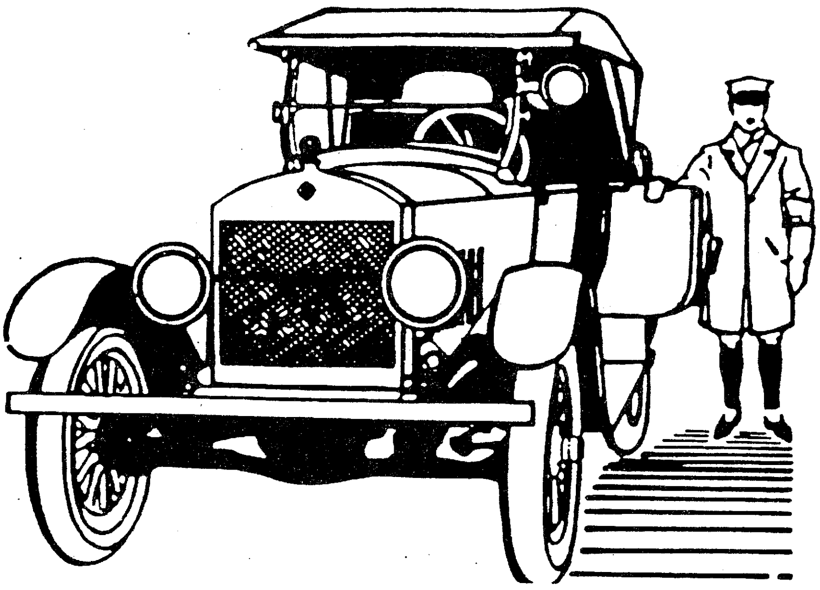


Fig. 5. Original picture for the example in section 5.3.

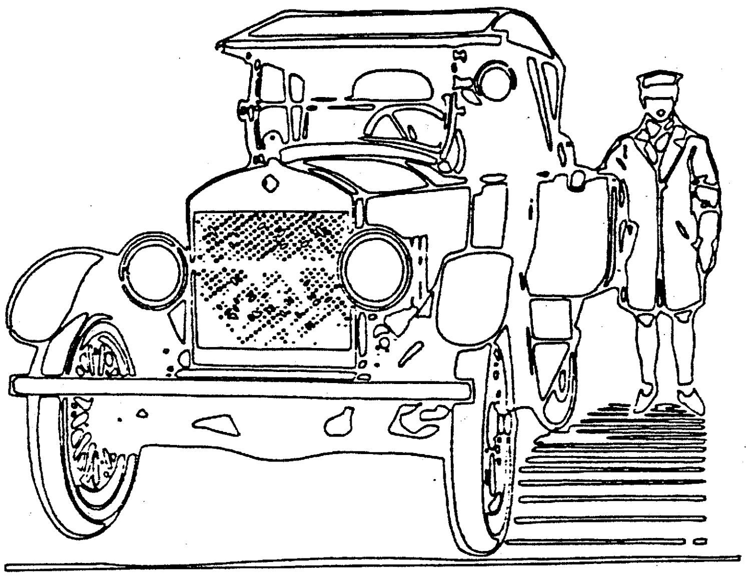


Fig. 6. Outlines of the picture in the preceding figure after the data reduction.

6. Conclusions

Two strategies for reducing the number of data necessary to approximate a given curve by Hermite–interpolation or by a B–spline curve, respectively, have been implemented and compared. The B–spline–based strategy has shown a clear superiority by allowing a better control of the smoothness and a further reduction of the number of knots. It was therefore included into a software package for the vectorization of the outlines of black and white pictures. Using this strategy saves a considerable amount of time, even though it still requires some post–processing by the user. Furthermore, the usefulness of the variable-speed parametrization technique could be illustrated.

An area for future research is to make the data reduction procedure from (Lyche and Mørken, 1987; 1988) taking into account conditions about the monotonicity or the (sign of the) curvature of the given curve such that e.g. the oscillations of the approximating curve in the lower part of the "B" in Figure 3 can be avoided.

References

C. de Boor (1978): *A Practical Guide to Splines.* — Berlin – Heidelberg – New York: Springer–Verlag.

E. Cohen and C. L. O'Dell (1989): *A Data Dependent Parametrization for Spline Approximation.* – In: Mathematical Methods in CAGD, (Eds. T. Lyche and L.L. Schumaker) Boston, pp.155–166.

G. Farin, G. Rein, N. Sapidis and A. J. Worsey (1987): *Fairing cubic B-spline curves.* — CAGD, No.4, pp.91–103.

J. Hoschek and D. Lasser (1989): *Grundlagen der Geometrischen Datenverarbeitung.* — Stuttgart: Teubner.

P. Karow (1986): *Digitale Speicherung von Schriften.* — Hamburg: URW–Verlag.

J. A. P. Kjellander (1983a): *Smoothing of cubic parametric splines.* — Computer Aided Design, No.15, pp.175–179.

J. A. P. Kjellander (1983b): *Smoothing of cubic parametric surfaces.* — Computer Aided Design, No.15, pp.288–293.

T. Lyche and K. Mørken (1987): *Knot removal for parametric B–spline curves and surfaces.* — CAGD, No.7, pp.217–230.

T. Lyche and K. Mørken (1988): *A Data–reduction Strategy for Splines with Application to the Approximation of Functions and Data.* — IMA J. Numerical Analysis, No.8, pp.185–208.

U. Wever (1989): *Darstellung von Kurven und Flächen mittels datenreduzierender Algorithmen.* — Ph.D.–thesis, TU München.

Received August 7, 1992
Revised December 10, 1993