A Limited Precision Method for Determining the Perimeter of a Flat Vector Object Using Bezier Curves

Dmitry Tarasov [1, a)] and Oleg Milder [1, b)]

[1]*Ural Federal University, Yekaterinburg, Russia*

[a)]Corresponding author: datarasov@yandex.ru
[b)]milder@mail.ru

Abstract. Plane shapes are one of the broadest domains for electronically stored information. Such vector objects, including texts, routes, etc. are often described using Bezier curves. Many data analysis tasks require determination of the perimeters of vector objects, which is associated with significant computational complexity; however, it is far from always necessary to calculate metrics with high accuracy. In this work, we propose splitting Bezier curves into arcs to reduce dimension. Thus, we quickly compute the perimeter of an arbitrary flat figure with limited precision.

Keywords: flat figure, Bezier curves, arc length, perimeter, splitter.

INTRODUCTION

The main type of information that is stored in electronic form is various types of text information containing an astronomical number of vector objects that make up it. Letters, routes, blueprints, vector illustrations, and the like are all domains of flat shapes that form a very complex subject of analysis and quantification. Quite often, there are problems of determining the perimeters of vector objects, for example, when calculating the irregularity.

Recently, textual information was evaluated only from semantic positions [1]. Today, a lot of attention is paid to its layout [2–4] and the calculation of spatial metrics, such as irregularity uniting the sum perimeter and area [5]. Any set of flat figures, especially those with a complex shape, expresses a certain complexity in defining these spatial metrics. The availability of a convenient method that allows such calculations to be made quickly and without the involvement of significant computing power would greatly facilitate the automated analysis of electronic information, which is typical for Big Data tasks.

In this article, we propose a method for calculating the perimeter of an arbitrary flat object in electronic form. To do this, we use the mathematical apparatus of Bezier curves, usually describing vector objects. In our approach, difficulties in calculating large dimensions (three and higher) are solved by splitting curves into arcs described by lower dimensions.

BEZIER CURVES IN VECTOR OBJECTS

Each electronic vector object (a character *etc*.) is formed by overlaying some Bezier curves. There are relatively simple methods for extracting outlines from the files [6]. The similar approach might be applied to the Bezier curve parameters in the PDF and PS/EPS files. One must note that these formats do not support the curves higher than the third order.

A Bezier curve is a parametric curve described by the corresponding polynomial and is a special case of the Bernstein polynomials [7]. In the vector graphics, the Bezier curves are used to model smooth curves that might be scaled to infinity. "Paths," as they are commonly called in imaging programs, are combinations of linked Bezier curves. In Fig. 1a, we show some common samples of the vector electronic objects. The star "*" is a simply

connected surface (space) that might be easily described by a series of the Bezier polynomials. The letter "a" and the ampersand show a complex cases of conjugate areas. This creates some difficulties in calculations due to the presence of inner and outer regions forming the shape of the objects.

A Bezier curve is defined by a set of control points P_0 through P_n, where n is called its order (n=1 for linear, 2 for quadratic, *etc*.). The first and last control points are always the *end* points of the curve; however, the intermediate control points (if any) generally do not lie on the curve. Given distinct points P_0 and P_1, a linear Bezier curve $P(t)$ is simply a straight line between them. For the cubic Bezier curve (1), four points $P_0(x_0, y_0)$, $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, and $P_3(x_3, y_3)$ in the plane are used, where P_0 is the "first" point, P_3 is the "last" point, P_1 and P_2 are managing ones. In the fonts, the curves are not self-intersecting that greatly simplifies the calculation of the curves parameters.

$$P(t)=(1-t)^3 P_0+3t(1-t)^2 P_1+3t^2(1-t)P_2+t^3 P_3,\Longleftrightarrow$$
$$\Longleftrightarrow\begin{cases} x(t)=(1-t)^3 x_0+3t(1-t)^2 x_1+3t^2(1-t)x_2+t^3 x_3 \\ y(t)=(1-t)^3 y_0+3t(1-t)^2 y_1+3t^2(1-t)y_2+t^3 y_3. \end{cases},t\subset[0;1]. \quad ()$$

Attempts to analytically calculate the arc length of a Bezier curve, the order of which is higher than the second, lead to the irrational integrals, in which, under the radical, there is a polynomial of high even degree that has no roots. Therefore, it is indecomposable into the prime factors on the field of real numbers. In this work, we propose an algorithm for the approximate calculation of the arc length of a Bezier curve of order higher than two. The algorithm is based on replacing the individual segments of the parent curve with the second-order Bezier curves that have the property that their arc length can be generally calculated through the coordinates of the control points. The arc length l of a smooth parametrized curve is given by the integral (2) where integration and differentiation is carried out with respect to the parameter t.

$$l=\int_0^1 \sqrt{\left(x_t^{'}\right)^2+\left(y_t^{'}\right)^2}\,dt. \quad ()$$

Consider how the arc length of the second-order Bezier curve might be expressed in terms of the coordinates of the control points. The general form of the second-order Bezier curve on the Cartesian plane is given by expression (3), where $P_0(x_0, y_0)$, $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ are the control points coordinates; P_1 is a managing control point and P_0, P_2 are the start and end points respectfully. Consider a parametrized second-order Bezier curve in general form (3) and transform it to the form of a polynomial with respect to the powers of the parameter t (4). Considering the introduced notation (5), we may show (6). Then the integral (2) takes the form (7).

$$P(t)=(1-t)^2 P_0+2(1-t)t\,P_1+t^2 P_2\Longleftrightarrow$$
$$\Leftrightarrow\begin{cases} x(t)=(1-t)^2 x_0+2(1-t)t\,x_1+t^2 x_2, \\ y(t)=(1-t)^2 y_0+2(1-t)t\,y_1+t^2 y_2 \end{cases} t\subset[0;1]. \quad ()$$

$$P(t)=t^2(P_0-2P_1+P_2)+t(-2P_0+2P_1)+P_0. \quad ()$$

$$A=(P_0-2P_1+P_2);\ B=(-2P_0+2P_1);\ C=P_0;\Longrightarrow P(t)=At^2+Bt+C;\ P_t^{'}=2At+B. \quad ()$$

$$x(t)=A_x t^2+B_x t+C_x,\quad y(t)=A_y t^2+B_y t+C_y,$$
$$x_t^{'}=2A_x t+B_x,\qquad y_t^{'}=2A_y t+B_y, \quad ()$$

¿

$$l=\int_0^1 \sqrt{4\,¿¿¿¿}. \quad ()$$

Note that the square trinomial under the radical in the integral (7) is certainly indecomposable into the simplest factors. We will not consider the degenerate cases: the point in the case when the integrand (2) is identically equal to zero, and the segment when the coefficients A in equations (6) are equal to zero. In these cases, integral (7) is taken easily. We introduce the notation (8). Moreover, the discriminant of the resulting square trinomial with respect to the parameter t is strictly negative. Taking into account the introduced designations (8), integral (7) is reduced to a tabular form and may be expressed as (9).

$$a=4\left(A_x{}^{¿¿}2+A_y{}^2\right);b=4\left(A_xB_x+A_yB_y\right);c=B_x{}^2+B_y{}^{2¿}. \quad ()$$

$$l=\int_0^1\sqrt{at^2+bt+c}\,dt\equiv\int\sqrt{X}\,dt,$$

$$\int\sqrt{X}\,dt=\frac{(2at+b)\sqrt{X}}{4a}+\frac{1}{2K\sqrt{a}}Arsh\left(\frac{2at+b}{\sqrt{D}}\right)+Const, \quad ()$$

$$X=at^2+bt+c;D=4ac-b^2;K=\frac{4a}{D};Arsh\,Z=\ln\left(Z+\sqrt{Z^2+1}\right).$$

EXPERIMENTAL AND RESULTS

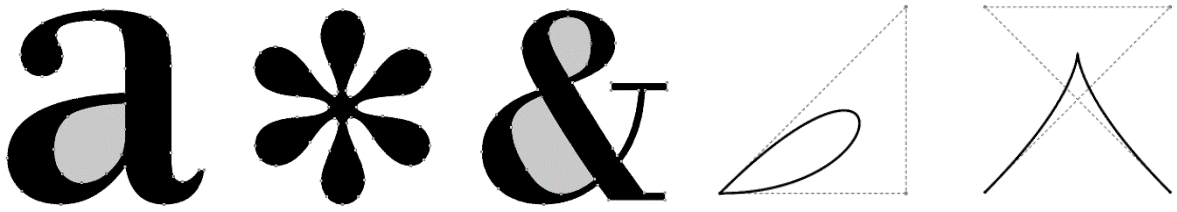
In order to evaluate our approach accuracy, we engage the relevant data on the Bezier curve arc length calculations by the elliptic integrals [8]. Here, we must note the influence of the human vision on our results. On the viewing distance (about 40 cm) with normal vision, the resolution of human eye is about 100 microns [9]. Considering the minimal object size to evaluate of 1 mm, we may deduce that relative precision 0.1 of the arc length calculation is enough for almost all the real-life applications.

In our approach, we first split an object (a letter *etc.*) into elementary curves (arcs) that make an object contour up and define the sets of control points, which we use further. The result of splitting is a set of 4×2 matrices, where 4 is the number of the control points for the third-order curve, 2 is the pair (*x*, *y*) of each control point flat coordinates. The number of such matrices in the set corresponds to the number of elementary curves, into which the contour of the object is splitted including the internal outline elements. Further we take each individual curve from the set. Note that in real contours the cases considered below, namely: self-intersection of the contour, cusp (sharp return) and self-closing are not present. At the first stage, the elementary curve is tested for the presence of zero curvature points. They are the real solutions of equation (10), which, in the case of the third-order curve, is reduced to a quadratic equation for the curve parameter *t*. Possible solutions for equation (10): there are no real roots in the given range of parameters, *i.e.,* arcuate or self-intersecting curve; one real root on the interval [0, 1], *i.e.,* s-shaped curve; two different real roots, *i.e.,* a curve with a smooth peak; one double root, *i.e.,* cusp point or sharp peak.

$$x_t'y_{tt}''-x_{tt}''y_t'=0,t\in[0;1]. \quad ()$$

Upon detection of zero curvature points, the primary ("parent") curve is splitted in accordance with the *de Casteljau*'s algorithm into segments by points of zero curvature. Depending on the number of roots in equation (10), the parent curve might be transformed into a set of at most three segments. Thus, one 4×2 matrix might be replaced by three matrices of the same dimension. In this case, each of the selected segments of the "parent" curve is considered as an independent ("child") curve with its own set of control points. At the second stage, each “child” curve is splitted to *N* parts by equal segments of the curve parameter *t* as the *De Casteljau*'s algorithm does not change either the order of the curve or its spatial position. Such, we propose, in approximate calculations of the arc length of a plane curve, to use not chords (segments), but parabolic segments. There is an algorithm described in [10] that allows one to increase the order of the curve, while maintaining its spatial position. Obviously, the inverse transformation can only be considered as an approximation. In particular, in relation to transformations between the Bezier curves of the second and third order, formulas (11, left part) were used. From these exact transformation formulas (where the superscripts in parentheses denote the order of the curve, and the subscripts are the indices of the control points) the formula for the approximate inverse transformation (11, right part) follows (only for *x*, for *y* the formula is the same).

$$\begin{aligned}x_1^{(3)}&=x_0^{(2)}+\frac{2}{3}\left(x_1^{(2)}-x_0^{(2)}\right)\\ x_2^{(3)}&=x_1^{(2)}+\frac{1}{3}\left(x_2^{(2)}-x_1^{(2)}\right)\end{aligned}\Longleftrightarrow x_1^{(2)}=\frac{3\left(x_1^{(3)}+x_2^{(3)}\right)-x_0^{(3)}-x_3^{(3)}}{4} \quad ()$$



(a) (b) (c)

FIGURE 1. (a) Objects built with the Bezier curves (nodes of the Bezier curves are shown): the letter "a" and "&"as non-simply connected surfaces; "*" as a simply connected one; (b) "egg", (c) "cusp" are the model objects for calculation perimeter as a sum of arc lengths, dotted lines are auxiliary elements constructed at the Bezier control points.

The notion "approximate" is used in the sense that replacing a set of control points with an increase in the order of a curve preserves the curve, increasing only the order of its analytical description. However, the inverse transformation being analytically accurate changes the shape of the curve. In order to estimate the lower bound of model accuracy, we add a simplest way to calculate the arc length by its chord. Each "child" curve might be further splitted into N equal parts as regard to the parameter t. Obviously, increasing N leads to increasing accuracy. The issue is what the minimum of N. As model objects, we engage two shapes, the "egg" and the "cusp" [11] (see Fig. 1b,c), for which the total arc lengths are calculated. Table 1 shows the results of calculations by mean of parabolic equation (11) and by linear chords evaluation in comparison with the data from [11].

Table 1. Perimeter calculation for the model objects, relative units; coincide values are in bold.

Split (N)	"Egg" Linear (chord)	"Egg" Parabolic	"Egg" Elliptic [11]	"Cusp" Linear (chord)	"Cusp" Parabolic	"Cusp" Elliptic [11]
1	0	not possible	18.3557	18.0278	18.3627	18.2843
2	16.7705	18.5426	18.3557	18.2050	18.2946	18.2843
3	17.2618	18.3285	18.3557	18.2466	18.2902	18.2843
4	17.7635	18.3614	18.3557	18.2625	18.2852	18.2843
5	18.0031	18.3356	18.3557	18.2701	18.2847	18.2843
6	18.0852	18.3556	18.3557	18.2743	18.2845	18.2843
8	18.2077	18.3556	18.3557	18.2786	**18.2843**	18.2843
10	18.2622	**18.3557**	18.3557	18.2806	18.2843	18.2843
12	18.2909	18.3557	18.3557	18.2817	18.2843	18.2843
15	18.3142	18.3557	18.3557	18.2827	18.2843	18.2843
20	18.3323	18.3557	18.3557	18.2834	18.2843	18.2843

CONCLUSION

Determining the perimeter of an arbitrary flat figure described by the third-order Bezier curves is a certain computational problem. We propose to solve this problem by splitting the curve into segments described by curves of the second order, for which the computational methods are well developed. The proposed method provides accuracy sufficient for calculating the perimeters of flat objects perceived by the human eye. The accuracy might be improved by increasing the number of splits; however, even splitting to 10 arcs gives satisfactory results. Thus, split to N=10 arcs gives the same results to parabolic approach (11) and to the far more complicated elliptic integrals. Moreover, even linear method (chords) gives satisfactory precision for the text-related area, *i.e.,* when evaluating perimeters of letters, one may use the arc splitting to 8–10 sub-arcs, which gives adequate evaluation.

REFERENCES

1. G. Amir, H. Murtaza, "Big data concepts, methods and analytics". International Journal of Information Management, 2015, 35, p.140.
2. M.D.S. Lonsdale, "The effect of text layout on performance: A comparison between types of questions that require different reading processes", Information Design Journal, 2015, 21(3).
3. R. Brath, E. Banissi, "Using Typography to Expand the Design Space of Data Visualization", The Journal of Design, Economics, and Innovation, 2016, Vol.2, Iss.1, 59–87.

4. D.A. Tarasov, A.P. Sergeev, V.V. Filimonov, Legibility of textbooks: a literature review, Procedia - Social and Behavioral Sciences, 2015, 174, 1300–1308.
5. D. Tarasov, A. Sergeev, “Irregularity as a quantitative assessment of font’s drawing and its effect on the reading speed”. CEUR Workshop Proceedings. 2015. Vol.1452. 177–182.
6. M. Sarfraz, M.F.A. Razzak “An algorithm for automatic capturing of the font outlines”, Computers and Graphics, 2002, Vol.26, Iss.5, 795–804.
7. B. Casselman “From Bézier to Bernstein”, 2006, feature column from American Mathematical Society http://www.ams.org/publicoutreach/feature-column/fcarc-bezier#2
8. H.Henkel “Calculating the Cubic Bezier Arc Length by Elliptic Integrals”, 2014, derived from http://www.circuitwizard.de/metapost/arclength.pdf
9. D. Tarasov “Vision and reading”, Ekaterinburg: UrFU, 2015. 76p. (Д. А. Тарасов, Зрение и чтение: монография – Екатеринбург: УрФУ, 2015. – 76 с. – in Russian).
10. G. Farin “Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide”, Academic Press, 1997, 473p.