# Variational-Methods-Based Glyph Vectorization

V. Kluzner[*], A. Tzadok[*] and M. Stanich[§]

Keywords: Bezier, contour, conversion, curve, raster

## Abstract

We propose here an alternative solution for font scaling, consisting of automatic glyph conversion from raster to outline type, successive enlargement of outline-type glyph, and eventually, rasterization of enlarged outlined type. This paper shows an approach for converting bitmap images of text glyphs into a vector format. The following work is concentrated on the first part of the algorithm (glyph vectorization). We formulate the above problem by variational means, where the input is the contour of the given glyph, consisting of the straight lines, and the output is the family of the third-order Bezier curves, outlining the given glyph and preserving its contour form.

## Introduction

The quality and speed of the printing industry is constantly improving. One major direction of these quality enhancements is increasing printing resolution. However, bi-level resources, such as raster fonts and logos encountered in legacy applications, may scale poorly to higher resolutions. It is difficult to scale these resources correctly without resulting in various visual artifacts such as jaggies or other distortions.

There exist various font enlargement methods, such as standard upscaling methods (e.g., bilinear and bi-cubic interpolation) or more complicated heuristic methods (Anderson (1986), (1987), (1989)). Because of the bi-level form of a given bitmap font, these methods do not preserve the form of the glyph. This problem exists also in the case where the scaling is an integer multiple.

We have checked the performance of one of these algorithms (Anderson (1989)), dealing with the 3 times enlargement of binary images stored in the run end form. The basic idea was to rotate the image by 90 degrees, to enlarge it vertically, to rotate back and to enlarge vertically again. The vertical enlargement included the interpolation of the two new lines between every two

[*] IBM, Haifa Research Lab, [kvladi, asaf]@il.ibm.com
[§] InfoPrint Solutions Company, stanich@us.ibm.com

horizontal lines, where the challenge  is determining when run ends correspond and detecting a variety of special cases which may require special handling. The above variety of cases included not only the relative run ends position of the given two lines, but also the run ends position of two additional lines adjacent to the two given lines from above and from below.

To explore the performance of the above algorithm, the Times New Roman size 24 text was enlarged three times and then compared to the same Times New Roman text of size 72 (see **Error! Reference source not found.**).

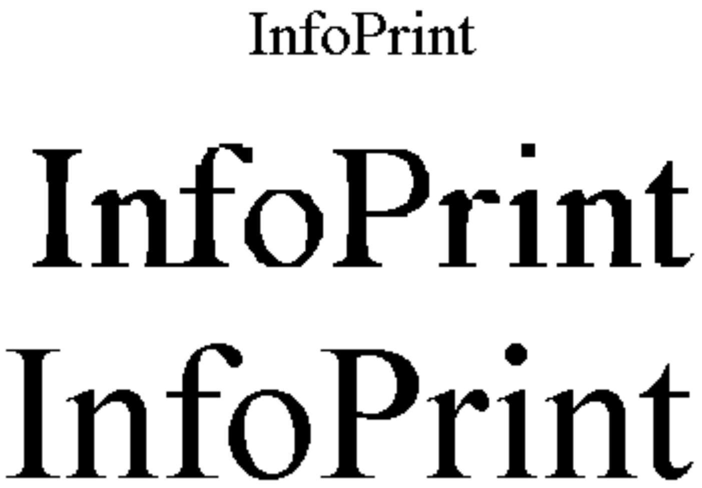


**Figure 1**
Size 24 text on the top, the same text 3 times enlarged in the middle and 72 size text at the bottom.

One may see that there are some abnormalities in the above enlargement process, for example, the "steps" on the vertical part of the first letter "I", and not exact glyph form conservation in case of letters "f" and "r", especially in their upper parts. One of the reasons in the above algorithm failures is the fact that the described interpolation should be performed in both dimensions (vertical and horizontal) simultaneously, because this action is not separable. Another reason for algorithm failure is an optional change in original lines that are used for interpolation. Due to above abnormalities we conclude  that there is a need in algorithm improvement.

Especially for glyphs it is desirable to obtain the same visual quality for different sizes and resolutions. When using bitmap fonts, the shape must be defined for each applicable size individually to guarantee optimal results. Moreover, bitmap images are consuming much more resources for storage and transmission. The glyph conversion to vector-based format seems to be the better choice for this kind of use cases.

This paper describes an alternative approach for upscaling raster font images for the non-integer and integer scaling cases. We propose a scaling algorithm that consists of the following three sections:

1. Glyph conversion from raster to outline (vectorization)
2. Enlargement of outline-type glyph
3. Enlarged glyph conversion from outline to raster (rasterization)

This idea was explored previously in Pletschacher (2006). The main drawback of their method was polygonal approximation of glyph contour. As a result, being scaled 4 times, the glyph contours looked polygonal and not "curved" enough. The need to use curves instead of straight lines became evident.

The main challenge, and the goal of this work, is to realize the first part of the suggested algorithm. Roughly speaking, we explore an algorithm that accepts the connected component (binary image) as input and produces its vector form, i.e., providing the number of curves outlining the given image. In this work, we use third-order Bezier curves for outlining the contour of a given glyph.

In the mathematical field of numerical analysis, a Bezier curve is a parametric curve important in computer graphics. Bezier curves were widely publicized in 1962 by the French engineer Pierre Bezier, who used them to design automobile bodies. The curves were first developed in 1959 by Paul de Casteljau using de Casteljau's algorithm, a numerically stable method to evaluate Bezier curves. In case of third-order (cubic) Bezier curve we have four points $P_0, P_1, P_2, P_3$ in the plane or in three-dimensional space defining it. The curve starts at $P_0$ going toward $P_1$ and arrives at $P_3$ coming from the direction of $P_2$. Usually, it will not pass through $P_1$ or $P_2$; these points are only there to provide directional information. The distance between $P_0$ and $P_1$ determines "how long" the curve moves into direction $P_2$ before turning towards $P_3$. The parametric equation form of the curve is:

$$C(t) = P_0 \cdot (1-t)^3 + P_1 \cdot 3t(1-t)^2 + P_2 \cdot 3t^2(1-t) + P_3 \cdot t^3,\ 0 \le t \le 1, \qquad \textbf{(1)}$$

see, for example, Figure 2.

Polynomial representation of a glyph outline, such as Bezier curves, is often used by font developers. Various examples include the TrueType fonts developed by Apple (see, for example, "d" glyph on Figure 3) or the Type 1 and Type 3 fonts developed by Adobe. We propose using this type of representation as an intermediate description to upscale existing bitmap fonts. As mentioned above, the main disadvantage of existing scaling algorithms is their distortion of the glyph being enlarged. We claim that outlining the binary image with a number of curves and its subsequent scaling preserve the glyph form during this enlargement process.

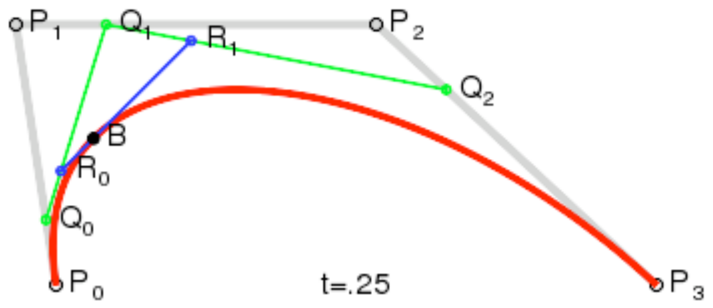


**Figure 2**
3rd order Bezier curve.

The question is, how can we find this family of curves outlining the given glyph, given the contour of the above glyph? Let consider our objectives for these curves. At first, they should be close to the given contour in some sense. Secondly, they should be of Bezier form and there should be a sufficient smoothness at the connection points. In addition, their length and their curvature should be restricted in some sense also. In order to incorporate these objectives, we use in this work the variational methods to formulate the above problem.

Let us consider, as a classical example of variational formulation, the Mumford-Shah variational model Mumford (1989) that has become a general framework in case of image segmentation. It defines the segmentation problem as a joint smoothing/edge detection problem as follows: given an image $g$, it seeks simultaneously a set $K$ of discontinuities, the "edges" of $g$, decomposing the planar domain $B$ ("pixel space") to disjoint connected open subsets $B_i, i = 1,\ldots,n$, so that $B = B_1 \cup \ldots \cup B_n \cup K$, and a function $u$ differentiable on $B = \cup B_i$, which is allowed to be discontinuous across $K$. Then, the "best" segmentation of a given image is obtained by minimizing the functional

$$F(u,K) = \alpha \int_B (u-g)^2 dxdy + \int_{B \setminus K} |\nabla u|^2 dxdy + \beta |K| , \qquad \textbf{(2)}$$

where $|K|$ stands for the total length of the arcs making up $K$ and $\alpha, \beta$ are two weight parameters. The smaller $F$ is, the better $(u,K)$ segments $g$. The first term accounts for how good is the approximation of $g$ by $u$. It is referred to as *fidelity* term. The second, *smoothness* term, makes sure that $u$ - and hence $g$ -

does not vary much on each $B_i$. The third term is a penalty on the total length of the discontinuities.

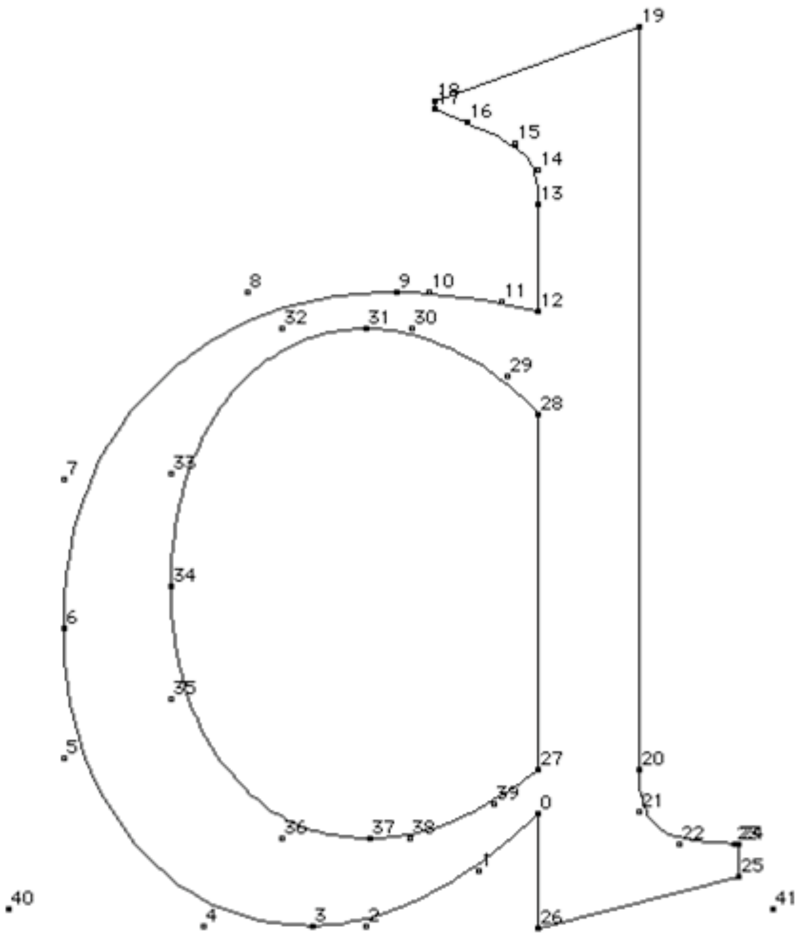


**Figure 3**
A TrueType "d" glyph "d".

Roughly speaking, we may summarize the variational formulation of a problem in the following informal way: given the data $g$, we want to find the output $u$ being close to $g$ in some sense. In addition it should satisfy some prior knowledge that we have about it. It is obvious, that the first part relating to the closeness to $g$, is being formulated in the fidelity term, and the prior knowledge about the desired output $u$ should be formulated in smoothness, or penalty term. Thus, for our problem we may generalize the Mumford-Shah functional (2) in the following way:

$$F(C) = \mathrm{Pr}ior(C) + \gamma \cdot Dist(C, g)\,, \tag{3}$$

where $C = (C_1, \ldots, C_N)$ is the family of desired third-order Bezier curves, $\mathrm{Pr}ior(C)$ is the penalty, or prior term and $Dist(C, g)$ is the fidelity, or distance term of the functional (3).

Our main innovation, however, lies in an appropriate formulation of prior and distance terms of the functional (3) and finding the outlining curves $C$ given the glyph contour $g$, in an accurate and visually pleasing manner.

## Variational formulation

Let $g(s)$ denote the contour of a given glyph, such as the glyph "c" in 24-size Times New Roman font shown in Figure 4.

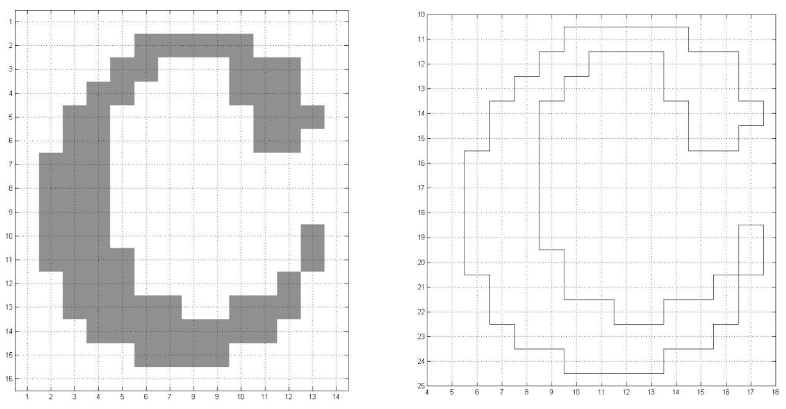


**Figure 4**
The glyph "c" and its contour.

We would like to find $N$ Bezier curves

$$C_i(t) = (1-t)^3 P_0^i + 3t(1-t)^2 P_1^i + 3t^2(1-t)P_2^i + t^3 P_3^i,\ \ 0 \le t \le 1,\ \ i = 1, \ldots, N \tag{4}$$

outlining the given contour. Thus, we are going to formulate formally the prior and distance term of the functional (3).

Actually, we have no special requirements for desired third-order Bezier curves. Our objectives are very common: we would like to restrict the length of these curves and their optional curvature (see, for example, Kass (1988)). In addition, we expect the continuity of the first and second order, smoothness of the zero and first order, for the global curve in locations of curves' connections. Our prior term looks, hence, as follows:

$$\begin{aligned} \Pr ior(C) &= \alpha \sum_{i=1}^{N} \int_0^1 \left| C_i'(t) \right|^2 dt + \beta \sum_{i=1}^{N} \int_0^1 \left| C_i''(t) \right|^2 dt \\ &+ \varepsilon_1 \sum_{i=1}^{N} \left( C_i'(1) - C_{i+1}'(0) \right)^2 + \varepsilon_2 \sum_{i=1}^{N} \left( C_i''(1) - C_{i+1}''(0) \right)^2 \end{aligned} . \quad \textbf{(5)}$$

The first term of the functional addresses curve length where the bigger $\alpha$ is the shorter the desired curve is. The second term deals with curve curvature, where the bigger $\beta$ leads to a straighter resulting curve. The third and the fourth terms address the continuity of first and second order, respectively.

The definition of distance term is a little bit more difficult. We cannot use here the $L^\infty$ norm or $L^2$ norm, as in case of Mumford-Shah functional (2): we deal here with one-dimensional manifolds in plane and not with functions of one variable which are by definition one-to-one correspondences. Our definition of distance between two curves is quite intuitive and logically motivated, as illustrated in Figure 5. Let us start from the distance from point $C$ to the curve $g(s)$, where $s \in S$ ($S$ is parameter domain). The intuitive definition of such distance should be the length from the point $C$ to the closest point on the curve $g$ and its formal definition is

$$Dist(C,g) = \inf_{s \in S} \left| C - g(s) \right| . \quad \textbf{(6)}$$

Assume now that we have a variety of such points in the form of the curve $C(t)$. How should we define the distance now? Considering that we would like to minimize the desired distance, the intuitive and logically motivated definition between two curves $C(t)$ and $g(s)$ should be

$$Dist(C,g) = \sup_{t \in T} \inf_{s \in S} \left| C(t) - g(s) \right| . \quad \textbf{(7)}$$

It should be noticed, that the above function is not commutative: in general, $Dist(C,g) \neq Dist(g,C)$ (see Figure 5). That is, minimizing the above distance we don't guarantee the closeness of two curves. Thus, the desired distance should include also its symmetric part, resulting in the final formula (8).

$$Dist(C,g) = \gamma_1 \sum_{t \in T} \inf_{s \in S} \left| C(t) - g(s) \right|^2 + \gamma_2 \sum_{s \in S} \inf_{t \in T} \left| C(t) - g(s) \right|^2 . \quad \textbf{(8)}$$

Both distances have been squared to simplify the following minimization. For the same reason we have changed the supremum function to a trivial sum of the distances. Assigning different values of $\gamma_1$ and $\gamma_2$, we can combine the two metrics in the global one, which will be satisfy to our needs.

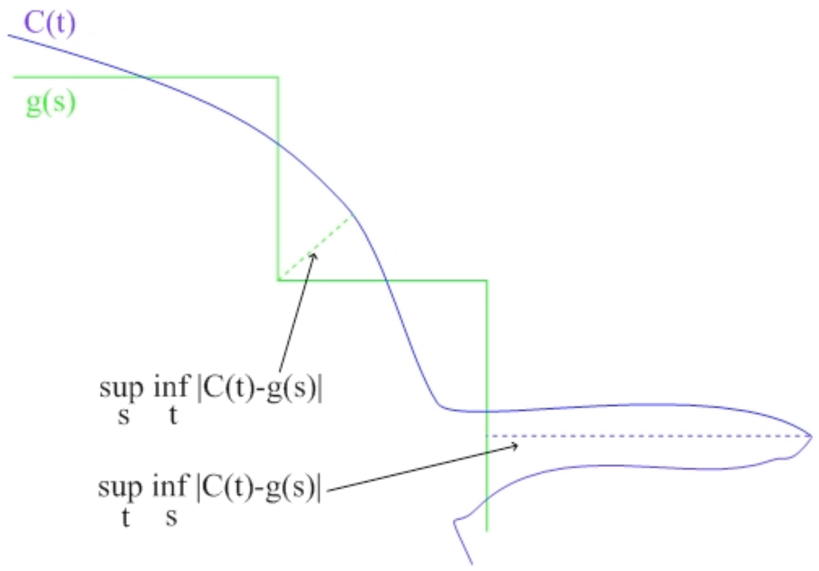


**Figure 5**
The distance between two curves in the plane

Our functional model is defined, thus, in the following way:

$$\begin{aligned}\Pr ior(C) = \alpha \sum_{i=1}^{N} \int_0^1 \left|C_i'(t)\right|^2 dt + \beta \sum_{i=1}^{N} \int_0^1 \left|C_i''(t)\right|^2 dt + \varepsilon_1 \sum_{i=1}^{N} \left(C_i'(1) - C_{i+1}'(0)\right)^2 \\ + \varepsilon_2 \sum_{i=1}^{N} \left(C_i''(1) - C_{i+1}''(0)\right)^2 + \gamma_1 \sum_{t\in T} \inf_{s\in S} \left|C(t) - g(s)\right|^2 + \gamma_2 \sum_{s\in S} \inf_{t\in T} \left|C(t) - g(s)\right|^2\end{aligned} \quad \textbf{. (9)}$$

We assume that finding the functional minimum will bring us to the desired solution.

## Minimum finding algorithm

We now turn to the desired form (4) of the curve $C(t)$. Roughly speaking, the functional variable is not the general curve $C$, but $3N$ points $P_1, P_2, P_3, ..., P_{3N-2}, P_{3N-1}, P_{3N}$, which are the Bezier curves' control points. Thus, our problem is being transformed from the variational one to the problem of minimizing the function of $6N$ variables – where every point $P_i$ is actually the pair $(x_i, y_i)$.

Due to the form of the functional (9) we can assume that it is convex, locally at least, thus there exists a minimum point. Is the minimum point we are going to find the local minimum point or the global one? We can't answer this question, but we may claim that we do not need to be concerned about it. We only need to find the minimum point that is close to the given data, the glyph contour, whether it is local or global one.

Eventually, in order to find the minimum point we must solve the nonlinear equation

$$\nabla F(P)=0,\ P=\left(P_1,P_2,P_3,\ldots,P_{3N-2},P_{3N-1},P_{3N}\right). \qquad \textbf{(10)}$$

There are some existing numerical methods for solving these types of nonlinear equations. Due to the robustness and simplicity of implementation we will use the fixed point scheme described in Vogel (1996) to solve (10). Substituting (4) in (9), we obtain observe that the first four parts in the right-hand side of (9) have the quadratic form concerning all $P_i$, which implies $6N$ linear equations with $6N$ variables in case of (10). A reminaing problem are the two last, metric, terms. In case of $\inf_{t\in T}\left|C(t)-g(s)\right|^2$ for every $s$ we define the derivation by $P_i$ as $2\left|C(t_0)-g(s)\right|\frac{\partial C(t)}{\partial P_i}$, where $C(t_0)$ is the closest point to $g(s)$, calculated "manually" from the previous iteration. The second case is much more complicated, because it is not so simple to find the closest point $g(s_0)$ on the contour $g$ from the point $C(t)$. Here we use the relaxation technique developed in Kluzner (2007): we assume that the expression

$$-\gamma \ln \sum_{s\in S} \exp\left(\frac{-\left|C(t)-g(s)\right|^2}{\gamma}\right) \qquad \textbf{(11)}$$

approximates $\inf_{s\in S}\left|C(t)-g(s)\right|^2$ when $\gamma\to 0$. The derivation of this term is defined in manner identical to the previous case.

Thus, at every stage we obtain from (10) the linear system of $6N$ equations with $6N$ variables, which are $(x,y)$ coordinates of $3N$ Bezier curves' control points. We do not deal here with proving the existence of minimum of the functional (9) and, as a conclusion, the existence of unique solution of linear system described above, but the form of the functional implies it. This was proved further by numerical examples.

Due to the method employed, of developing the derivation of two metric terms, our solution has to be iterative. As we mentioned before, we tend to find the minimum point close to the initial data - in this case the given glyph contour. A problem occurs because the initial contour consists of vertical and horizontal lines. To describe it in terms of Bezier curves we first calculate a coarse approximation of the given contour in terms of Bezier curves. We perform this operation using a "curve growing" method: starting at arbitrary point on the given contour, we find the Bezier curve approximating the maximal part of this contour and also sufficiently close to it. This is shown, using, up to a 0.2 pixel difference, in Figure 6.

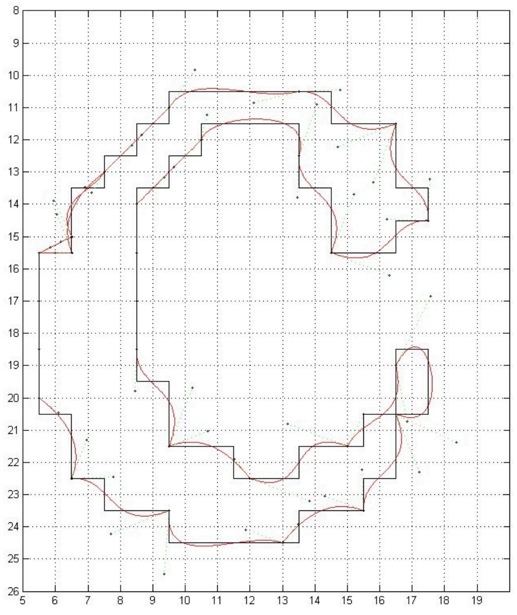


**Figure 6**
Initial approximation of the glyph "c" contour.

Now with the initial solution found, we start the iteration process. Empirically we found that the amount of 30 iterations is sufficient. The accepted family of Bezier curves outlines the original contour very similar to expected one as shown in Figure 7 on the left side.

At this stage we have to deal with two additional problems: the optimal number of Bezier curves and the location of Bezier curves' end points. These two issues are strongly connected. Analyzing the curvature graph on the domain of the global Bezier curve, shown on Figure 7 on the right side, we find the local maximum points where the curvature is locally maximal and place the new Bezier curves' end points at these locations. This technique is used by various font developers: the Bezier curves outlining the future glyph start and end at points with high curvature. Now the number of Bezier curves outlining the given contour is not arbitrary, but defined in some logical way. Secondly, placing the end points in places with high curvature allows in the future optional sharp points on the global outlining curve. This can be observed at the lower end of the glyph "c" on the Figure 8.

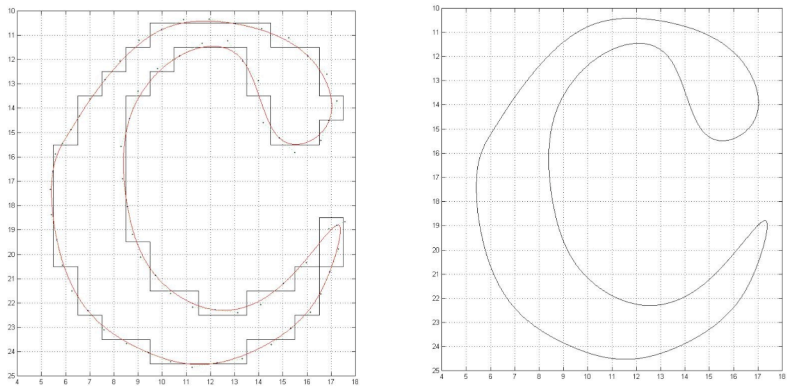


**Figure 7**
The intermediate approximation of the glyph "c" contour (on the left) and accepted global Bezier curve (on the right).

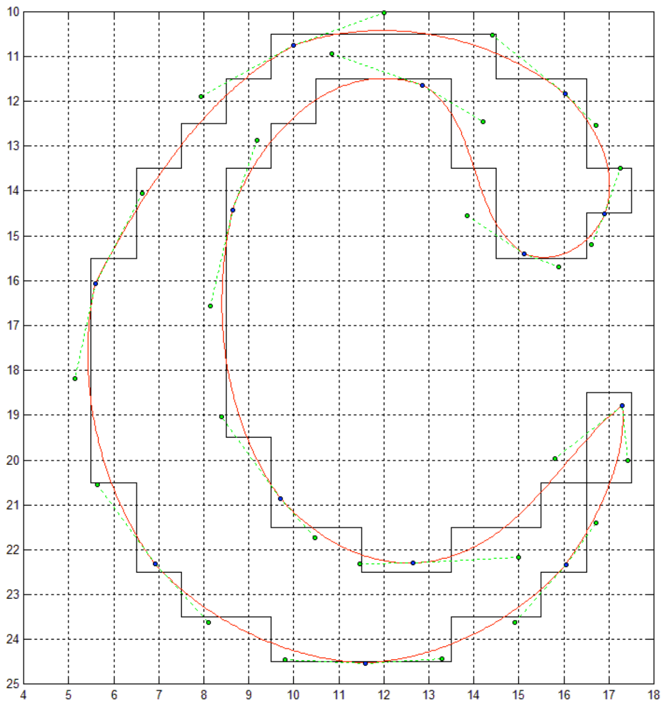


**Figure 8**

End points re-arrangement on the Bezier curve outlining the glyph "c" contour.

At this stage we continue the iterations, with the initial global curve after the end points' re-arrangement process. We slightly modify the prior term in our functional: at the new end points with extremely high curvature we omit the zero and first order smoothness. At the remaining end points we replace the zero order smoothness demand $C_i^{'}(1) = C_{i+1}^{'}(0)$ by curve velocity collinearity demand $C_i^{'}(1) = k \cdot C_{i+1}^{'}(0)$. The previous penalty was used for stabilizing the global curve form; now, when the form of the glyph has been sufficiently outlined the collinearity request is sufficient. The final glyph "c" contour outlining curve and its superposition on the initial glyph are presented on the Figure 9 and on the Figure 10 respectively.

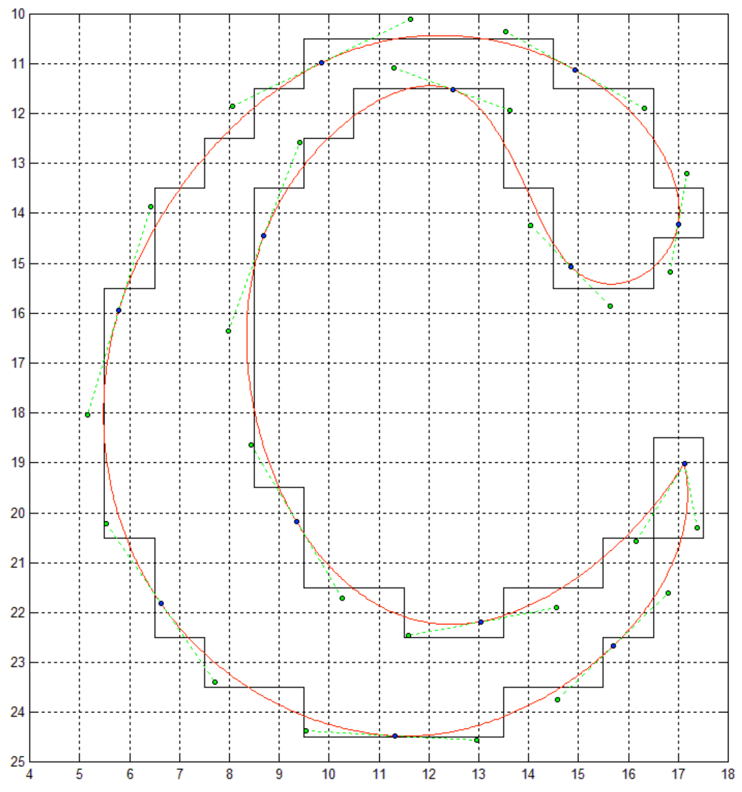


**Figure 9**

The final glyph "c" contour outlining curve.

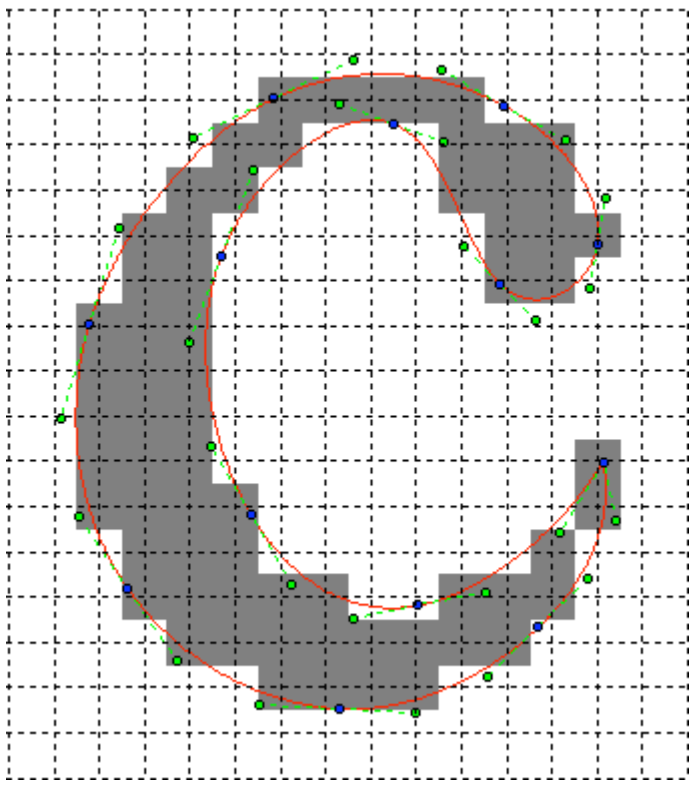
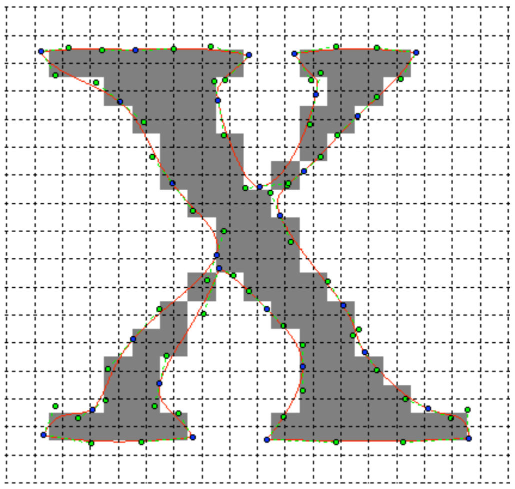


**Figure 10**
The outlining curve on the initial glyph.

## Additional examples and conclusions

As may be seen on the example of glyph "c", the results in smooth glyphs like "a", "o", "e", "d" etc. are ideal: the algorithm deals perfectly with smooth parts which have the form of circle and ellipse. The sharp parts in these glyphs are also outlined correctly. The main drawback of this algorithm is glyphs with serifs.

We present here the outlining solution for two additional glyphs "x" and "y" of the same type Times New Roman size 24. These two glyphs were chosen due to their "uncomfortable" form, which is expressed in serifs in their upper parts and in the lower part of glyph "x" (see Figure 11). In addition, the outlined glyph from Kanji font of size 64 is presented in the Figure 12. As may be seen, the results are quite good, except the lack of sharp corners in places, like the internal part of outlining curve of the lower part of glyph "x", or interior outlining of Kanji font which should be square with sharp corners. The problem that causes this is a non-ideal algorithm for choosing the end points on the outlining contour that should not be smooth. This is the main remaining issue that should be the part of further research.



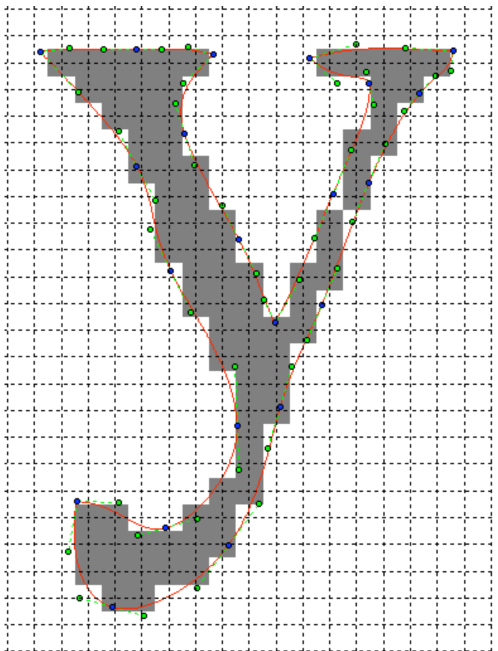


**Figure 11**
Outlined "x" and "y" glyphs.

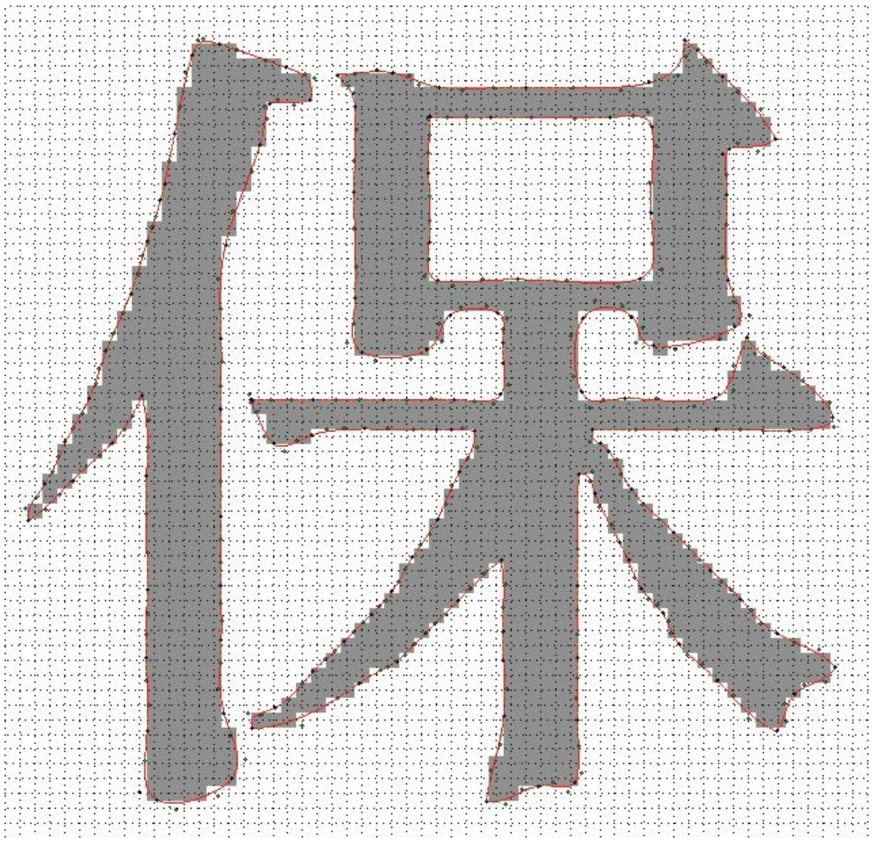


**Figure 12**
Outlined Kanji glyph.

## Acknowledgements

The authors would like to thank Guy Rosman from IBM Haifa Research Lab for valuable discussions during this work.

## Literature Cited

Anderson, K.L., Mintzer, F.C., and Goertzel, G.
1986 U.S. Patent 4,631,751 (December 23, 1986).

Anderson, K.L., Mintzer, F.C., Goertzel, G., Mitchell, J.L., Pennington, K.S. and Pennebaker, W.B. 1987 "Binary-image-manipulation algorithms in the image view facility", IBM J. Res. Develop., vol. 31, no. 1, pp. 17-31.

Anderson, K.L., Pennebaker, W.B., and Pennington, K.S. 1989 U.S. Patent 4,885,786 (December 5, 1989).

Kass, M., Witkin, A. and Terzopoulos, D. 1988 "Snakes: Active contour models", Internat. Journal of Comp. Vision, vol. 1, pp. 321-331.

Kluzner, V., Wolansky, G., and Zeevi, Y. Y. 2007 "A geometric-functional-based image segmentation and inpainting", in Proceedings of First International Conference on Scale Space and Variational Methods in Computer Vision, Ischia, Italy, May/June 2007.

Mumford, D. and Shah, J. 1989 "Optimal approximations by piecewise smooth functions and associated variational problems", Comm. Pure Appl. Math., vol. 42, pp. 577-685.

Pletschacher, S. Eckert, M. and Hübler, A.C. 2006 "Vectorization of glyphs and their representation in SVG for XML-based processing", Proceedings ELPUB2006 Conference on Electronic Publishing, Bansko, Bulgaria, June 2006.

Vogel, C. and Oman, M. 1996 "Iterative methods for total variation denoising", SIAM J. Sci. Statist. Comput., vol. 17, no. 1, pp. 227-238.