Representing contours as sequences of one dimensional functions

Sporring, Jon; Arps, Ronlad

Published in:
<em>Proceedings of the 4'th Asian Conference on Computer Vision (ACCV'00)</em>

Publication date:
2000

Document version
Early version, also known as pre-print

Citation for published version (APA):
Sporring, J., & Arps, R. (2000). Representing contours as sequences of one dimensional functions. In *Proceedings of the 4'th Asian Conference on Computer Vision (ACCV'00)* (pp. 742-746)

Representing Contours as Sequence of One Dimensional Functions

Jon Sporring[*](sporring@ics.forth.gr)
Institute of Computer Science, Foundation for Research and Technology – HELLAS
Vassilika Vouton, P.O. Box 1385, GR-71110 Heraklion, Crete, Greece

And

Ronald Arps (arps@almaden.ibm.com)
IBM, Almaden Research Center
650 Harry Road, San Jose, CA 95120-6099, USA

ABSTRACT

A sequence of one-dimensional geometric descriptions is suggested, for representing the contours of text characters (blobs) in black and white scanned textual images. This representation can be used to analytically encode alphabets of similar blobs into pseudo-fonts that are resolution independent, while simultaneously maximizing compression potential. Such a sequential one-dimensional representation was found to retain flexibility and to be at least twice as compact, when compared to two-dimensional contour representations classically used for representing generated resolution independent fonts.

Keywords: Contour Representation, Spline, Contour Segmentation, Minimum Description Length, Scale-Space.

1. Coding Office Documents

Electronic storage and transmission of scanned office documents is of increasingly important in office environments. The fax machine has already been used worldwide for decades, and with the maturing of the Internet it has become possible to make old paper documents, such as patents, electronically available.

The raw scanned format is a wasteful representation which is inefficient for direct storage and transmission: A full page is roughly 90 square inches, and if a scanning resolution of 600 dots per inch is used, this corresponds to $3 \cdot 10^7$ black and white pixels or 4 megabytes. Conversely, the main carrier of information for most office documents is the text, which only consists of approximately 5000 characters from an alphabet of 128 possible or about 2.5 kilobytes. Unfortunately, neglecting page layout and font sets by sending only non-formatted text seriously diminishes reading quality.

The work presented in this article is proposed to be part of a larger textual image compression system like JBIG2 [1], where an office document is represented in a lossy manner as the alphabet of blobs plus a list of blob codes and placements. We will also study the lossless compression of blob bitmaps from such an alphabet.

Several other lossy compression systems have been suggested to date. Systems based on frequency encoding such as the JPEG standard do not perform well on black and white images, since very high frequencies are present. Systems that try explicitly to identify the font sets and type also exist, but they have two main disadvantages: Firstly, the number of fonts grows day by day making the task increasingly difficult, and secondly, when the wrong font or type is identified, the result is found to be very disturbing to the human eye.

The most powerful lossless systems like JBIG2 are based on nearby pixel compression models. Such algorithms mostly disregard the geometrical content and compresses based on the statistics of the neighboring pixels. Such systems are highly successful in terms of compression, but they have some limitations. Firstly, they are essentially one dimensional and inherently partition the document into what has been read and what has not. Hence, the full two dimensional structure is neglected. Secondly, these systems do not use a model class that is close to what originally produced the data. Characters are often the result of a geometrical language, e.g. characters in the Postscript language and the MetaFont program are represented by a collection of polynomials. Finally, the model is not present in an analytical form, and requires multiple representations to decode at various resolutions.

We will here study a geometrical an analytical representation of scanned blobs. Blobs are representable by their closed contours, since the filling of the space between contours easily can be asserted by the odd/even fill rule: Examine any straight line on a page starting from a known color, and flip color each time a contour is crossed. The goal of this work is to code blobs in a lossless manner, by splitting the code into a geometrical model and a noise signal. It is the underlying intent to investigate the usability of the model alone as a lossy code.

The list of literature for geometrical descriptors of contours is extremely long, but we have especially found [2, 3, 4] to be valuable. It seems that the novelty of our work is to combine the myopic view of differential geometry with information theory to gain a connection between local and

*This work is part of a joint study program between the Department of Computer Science, University of Copenhagen, Denmark and IBM, Almaden Research Center. The major part of the effort was conducted at Almaden Research Center.

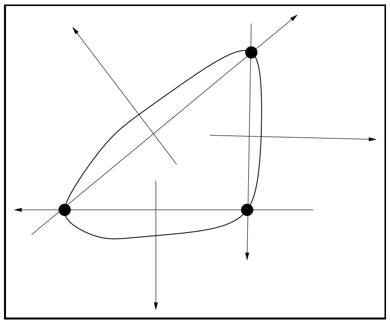

Figure 1. With the knowledge of the knots, a coordinate system may be placed such that the contour in between two knots is 1D. Black circles are knots, arrows are the three local coordinate systems, and the thick line is the contour to be approximated.

global models. Connecting local and global is not possible without comparison of description complexity with the error of the approximation. For this we will use the Minimum Description Length methodology [5].

2. 1+1D Contour Models

Many models for contours are based on modeling each of the two coordinate functions $x(t)$ and $y(t)$, where $[x(t), y(t)]^T$ gives the contour. This representation is however not rotational invariant, and not a very compact one either, since contours are essentially 1D structures.

The Fundamental Theorem of Curves states that any continuous 2D curve can be described by the curvature function up to a Euclidean movement [6]. In the (local) Frenet coordinate system this implies that the curves are locally well represented by the Frenet approximation,

$$\vec{x}(s) = \vec{x}(s_0) + s\vec{T}(s_0) + \frac{s^2}{2}\kappa(s_0)\vec{N}(s_0),$$

where $\vec{T}$ and $\vec{N}$ are the tangent and the normal vectors for the curve.

Although the curvature uniquely describes the curve, it might not be the computationally most feasible representation. One major draw-back is that in order to solve the Frenet approximation given a curvature function, one has to solve a differential equation and the errors accumulated thereby seem difficult to handle. Further, it is unclear how to incorporate the full knowledge of the limited set of curvature functions that generate closed non-intersecting curves.

An alternative approach is to cut the shape into pieces representable as 1D functions each with its own coordinate system. This is known as a Monge patch. An example of this is shown in Figure 1. The closed curve is cut into three pieces, say, and the positions of the cuts (knots) define local coordinate systems for each piece. For symmetry reasons, the origin of each local coordinate system has been place on the center point connecting two knots.

One obvious advantage of this representation is that the shape is studied as a one dimensional entity, and although it is not the intrinsic curvature function, each piece will converge towards the Frenet approximation as the density of knots tends to infinity. Further, if the knots are distributed according to the absolute curvature, the derivations from the intrinsic shape is greatly reduced. To see this, view the shape in the myopic perspective, i.e. by zooming until everything looks locally linear. Clearly, a representation by piecewise linear functions differs very little from the intrinsic shape in this perspective. The amount of zooming necessary is proportional to the curvature: When the curvature is large the zooming has to be great, and vice versa for small curvatures. The major disadvantage is that the cutting is a global process, i.e. each placement of a knot depends on the placement of the neighboring knots and the curve in between. Secondly, when only very few knots are used, the functions between knots are far from the intrinsic shape. E.g. a circle needs at least 2 cuts, while its curvature function is a constant.

To summarize, we will use a set of coordinate pairs and connecting 1D functions. Although we make use of 2D coordinates this representation is still rotational invariant since the majority of contour points are modeled as rotational invariant 1D functions, and the knots will be chosen in a rotational stable fashion to be described in the following. We call this the 1+1D model.

3. A Shape Algorithm

In order to approximate a contour by the 1+1D model, we need to identify a number of knots on the contour, and model the contour in between knots by 1D functions. To reduce the description length of a blob we further need to decide on the model complexity.

3.1. Knots and 1D Functions

Psychophysically, the semantically important points on a contour are those with extremal curvature (corners) [7]. However, the locations of extremal curvature points are sensitive to noise, hence the image and its derivatives must be regularized. For this we will use a scale-space [8, 9], in which an image is extended with a scale parameter such that the image structure is simplified as scale is increased. In this paper we will use the Linear scale-space, defined by the diffusion equation:

$$\frac{\partial}{\partial y}I(x,t) = \frac{\partial^2}{\partial x^2}I(x,t);\;\; I(x,0) = I(x)$$

where $I(x)$ is an any dimensional image, x is the spatial coordinate and t the scale coordinate. The Greens function of the diffusion equation is the Gaussian or normal kernel, implying that we can implement the diffusion equation by smoothing the original image with a Gaussian kernel.

Other scale-spaces are also available, and especially the Mean Curvature scale-space of curves [10] have been examined in this context [11]. However, for Internet applications it is important to be able to progressively decode blobs in a coarse to fine manner, which the linear scale-space models. We will therefore prefer this scale-space over others here, but we note that the Mean Curvature scale-space is the simplest to use if the resolution of the image is fixed.

Taking the set of corners as knots does not absolutely guarantee that the contour in between can be viewed as a 1D function. For example, a perfect circle has no corners and must be cut in at least two points, for it to be codeable as a sequence of 1D functions. Hence, we are forced to introduce extra knots. Several methods have been suggested in the Spline literature, and we choose to sample the integral of the square root of the absolute curvature linearly in between knots as suggested by de Boor [12]. This implies that extra knots are introduced when there is much curvature, which conforms with the myopic view as described earlier.

Hence the algorithm so far is as follows:

1. `Calculate a scale-space of the original image and find the contour of the blobs at each scale.`
2. `Calculate the curvature function and find extremal curvature point for each scale.`
3. `Track the evolution of extremal curvature points over scale.`
4. `Represent the curves between knots as a 1D function, adding extra knots where needed according to the integral of the square root of the absolute curvature.`

The algorithm has 3 parameters: The sampling density in (natural $\tau = \log t$) scale, a threshold for which extremal points are semantically important, and a parameter controlling the frequency of non-extremal knots.

Most importantly is the setting of the sampling density in the natural scale variable. This can fairly easily be set a priori depending on the desired precision of the corner tracking. Increasing t with 10% each step seems a reasonable setting in general. The threshold for extremal points depends more on the task and the noise level of the images. Finally, the Monge Patch approximation restricts the allowable constant increment in the integral of the square root of the absolute curvature. The smaller this value is, the better the approximation. It is clear that the curve must not turn more than 180°, and a reasonable guess is to disallow turns larger than 90°.

Both the semantical threshold and the frequency of non-extremal knots very much affect the computation time of the algorithm, since the computational most expensive part of the algorithm is the minimization of the descriptive complexity, which will be explained in the next section.

3.2. Choosing Model Complexity

An important concept in descriptive complexity is the Minimum Description Length (MDL) principle [5]. The MDL principle states that data should be described in the shortest possible fashion given a class of functions. An approximation of the MDL principle used often in the literature is,

$$\arg\min_{\delta,\theta} L(x) = L(x|\theta) + L_\delta(\theta), \qquad (1)$$

where x is a vector of data points, and θ is a vector of parameters identifying a model given a model class up to a given precision δ, $L_\delta(\theta)$ is the number of bits used to code the parameters, and $L(x|\theta)$ is the number of bits used to code the residual. The equation in (1) is only an approximation, since a specific estimator for θ limits the possible data sets x, which can have been the cause of the estimate θ. Hence $L(x|\theta)$ should be renormalized to reduce the code length [5]. A code of the contour may be designed using a great variety of models. We have performed experiments on model constraints such as first order continuity at the knots and progressive estimation of tangent on the knots. In this article we will present the simplest version, where a polynomial is fitted to the curve between knots such that the polynomial exactly passes through the knots and the squared distance is minimized. This implies zero order continuity but first order discontinuity at knots. Such a choice is reasonable for knots that represent corners. Further, since two degrees of freedom have been fixed, fitting a contour piece with a third degree polynomial implies fitting the remainder with a line using the method of weighted least squares.

The fitting of polynomial pieces only depends on the knots and not on the neighboring fits. Hence we may code a contour by first sending the set of knots followed by the set of polynomial parameters for each contour piece. The total code length of a contour is thus given as

$$L = \sum_i L(\text{knot}_i) + \sum_i L(\text{contour-piece}_i).$$

Since the code length of each contour-piece only depend on the knots we may complete our algorithm from Section 3 as follows:

5. `Calculate the code of knots` $\sum_i L(\text{knot}_i)$ `and for each contour-piece` i`, find the optimal model that minimizes` $L(\text{contour-piece}_i)$.
6. `Iteratively remove a knot yielding largest reduction in the total coding cost` L `until no further decrease can be found.`

The algorithm thus sketched is a split and merge algorithm, and we emphasize that finding the global minimum is intractable [13]. However a variant exists that uses the scale-space structure to perform local instead of global mergings resulting in a shorter computation time [14].

4. Coding an Alphabet

We have modeled 157 blobs by their contours taken from a standard CCITT fax image[†]. The blob sizes ranged from 5×5 pixel to 64×64 pixels. In order to compare model flexibility with the Postscript font encoding we choose to enforce that the models should meet in knots, and we fix the model class to a 2 parameter polynomial. This implies that each contour piece will be fitted with a local 3 order polynomial (4 degrees of freedom). For the 157 blobs we found 1193 polynomial pieces and an equal number of knots. That is an average of about 7.5 pieces per blob with the use

[†] `ftp://ftp.funet.fi/pub/graphics/misc/test-images`

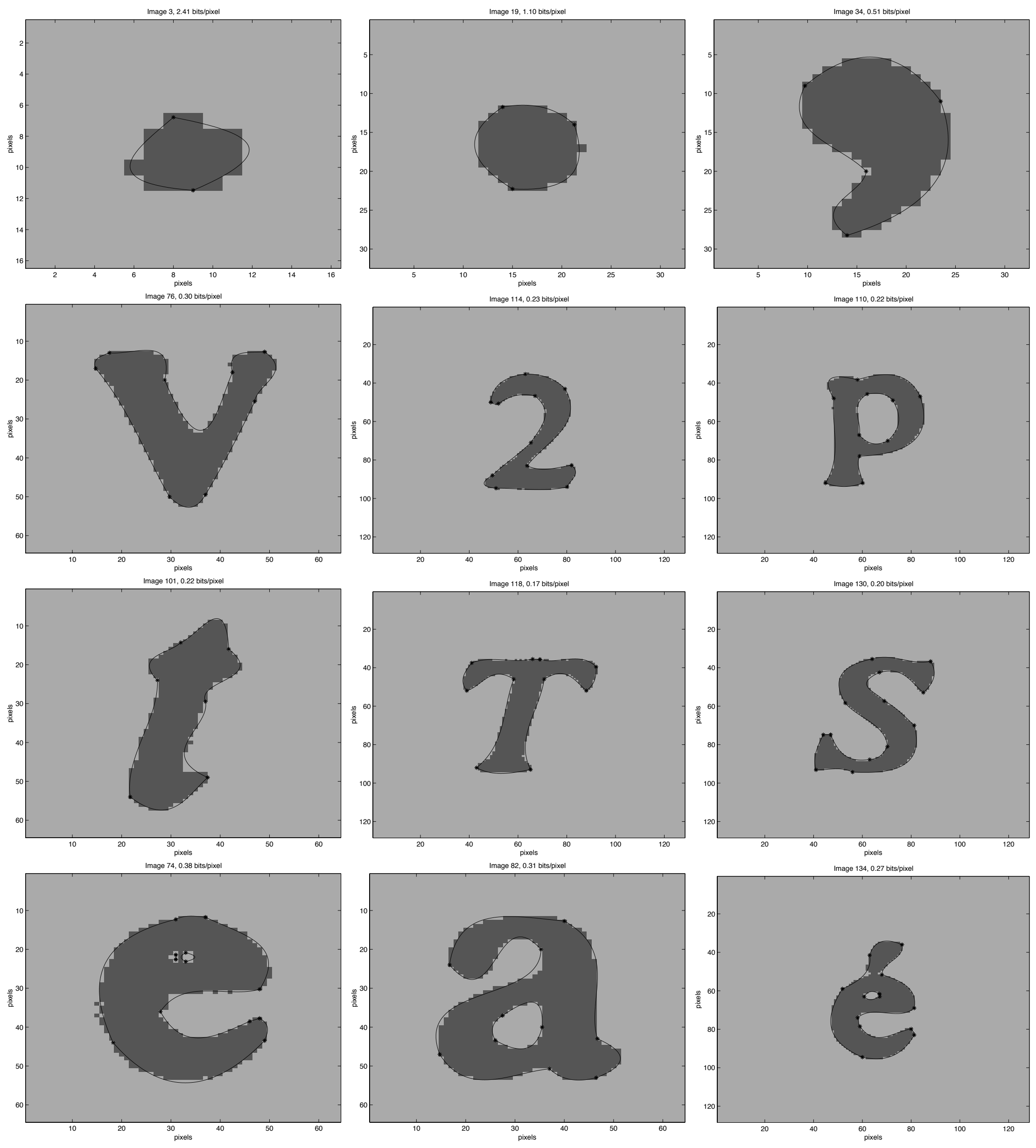

Figure 2. Sequences of one-dimensional functions easily capture the structure of typical blobs from office documents. The blobs in this figure have been selected such that the rows from top to bottom show: small blobs, large blobs, italic blobs, and worst experienced model correspondence.

of a total of 55957 bits (as estimated by the MDL functional). A representative selection is shown in Figure 2.

Visually we judge the resulting geometric models to be very good. We observe that the models use 3–4 knots to code a circular boundary, where 4 knots are used for large circles. There is a tendency that straight pieces are not coded as such, but we conclude that this is only natural, since straight pieces are not a generic part of the 2 parameter polynomial model class.

5. Blob Coding in Perspective

Models for describing blobs are a central issue in applications related to image storage or transmission. In this work, a novel model class has been suggested taking the one dimensional nature of blob borders into account. Experimentally this has been shown to yield both compact codes and good descriptors. Optimization is very much a part of finding good models within a class, and here we have presented a greedy thinning algorithm applied on a carefully selected set of knots. This seems to use a good tradeoff between compression time and ratio, and we conclude that this model class is useful in a system like JBIG2 [1] symbol bitmap coder.

To end, this work has shown that sequences of 1D functions is a feasible representation of typical blobs in office documents, and that it is possible to estimate such a representation from bitmaps. This has two implications.

We note that automatic conversion of scanned bitmaps into analytical representations is a useful tool:

- Since blobs can be decoded any resolution independently on original scanning resolution, the framework presented here could be used as an interface between low resolution originals and higher resolution printers.
- It is also possible to derive a resolution depended description on the described model class, which would be useful for progressive decoding of blobs in a coarse to fine manner by transmitting further knots.

Finally and most importantly, sequences of 1D functions may be used to refine the font technology of e.g. the Postscript language. We have demonstrated that our representation is independent on the position and orientation of the coordinate system, and that the representation uses only half the number of parameters while retaining the same functionality as Postscript fonts. Hence, this representation can be used to compress the very large Postscript font dictionaries without essentially limiting functionality.

References

[1] JBIG committee. Lossy/lossless coding of bi-level images, November5 1998. ISO/IEC Committee Draft Standard 14492 (JBIG2).

[2] Tony Lindeberg and Meng-Xiang Li. Segmentation and classification of edges using minimum description length approximation and complementary junction cues. *Computer Vision and Image Understanding*, 67(1):88–98, 1997.

[3] Paul L. Rosin and Geoff A. W. West. Nonparametric segmentation of curves into various representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1140–1153, December 1995.

[4] Mei-Hsing Chen and Roland T. Chin. Partial smoothing splines for noisy +boundaries with corners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1208–1216, November 1993.

[5] Jorma Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, January 1996.

[6] J. J. Koenderink. *Solid Shape*. M. I. T. Press, Cambridge, 1990.

[7] F. Attneave. Some informational aspects of visual perception. *Psychological Review*, 61(3):183–193, 1954.

[8] B. M. ter Haar Romeny, editor. *Geometry-Driven Diffusion in Computer Vision*. Number 1 in the series Computational Imaging and Vision. Kluwer Academic Publishers, Dordrecht, Netherlands, 1994.

[9] Jon Sporring, Mads Nielsen, Luc Florack, and Peter Johansen, editors. *Gaussian Scale-Space Theory*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[10] Farzin Mokhtarian and Alan Mackworth. Scale-based description and recognition of planar curves and two-dimensional shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):34–43, January 1986.

[11] Jon Sporring, Xenophon Zabulis, Panos E. Trahanias, and Stelios C. Orphanoudakis. Shape similarity by piecewise linear alignment. (submitted for publication), June 1999.

[12] Carl de Boor. *A Practical Guide to Splines*. Springer-Verlag, 1978.

[13] Martin C. Cooper. The tractability of segmentation and scene analysis. *International Journal of Computer Vision*, 30(1):27–42, 1998.

[14] Jon Sporring and Ole Fogh Olsen. Segmenting by compression using linear scale-space and watersheds. In *Scale-Space Theory in Computer Vision, Proc. 2nd International Conference*, Lecture Notes in Computer Science, Corfu, Greece, September 1999. Springer-Verlag.