SIGNAL PROCESSING IV: Theories and Applications
J.L. Lacoume, A. Chehikian, N. Martin, and J. Malbos (Editors)
Elsevier Science Publishers B.V. (North-Holland)

CURVE FITTING FOR THE RECOGNITION OF LINE DRAWINGS

V. Steinbiss, R. Ullrich, H. Ney

Philips GmbH Forschungslaboratorium Hamburg,
P.O. Box 54 08 40, D-2000 Hamburg 54, West Germany

This paper describes an approach to the recognition of line drawings and test results. The approach is based on fitting curve primitives such as straight lines and circular arcs to the lines in the image to be recognized in order to produce a higher level description of the image. This technique is being developed within the framework of automatic processing and recognition of line drawings in technical documents, such as mechanical construction drawings or freehand drawings.

1. INTRODUCTION

After scanning, the binary image of the line drawing is subjected to a standard thinning algorithm in order to construct the set of skeletal points. This skeleton of the image serves as the input for the two-step procedure studied in this paper. The first step is to decompose the skeleton into a set of segments, each of which defines a contour. In the second step, each segment is to be explained by a sequence of curve primitives. Fig. 1 summarizes the architecture of our system.

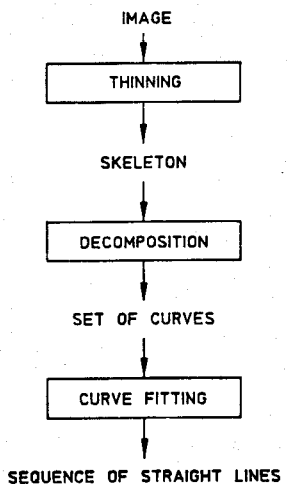

Figure 1
Illustration of system operations

2. SKELETON DECOMPOSITION

The aim of decomposing the skeleton is to obtain segments with a simple structure, less complexity and homogeneous line width. Each segment is delimited by two nodes. We define a set of connected segments as a graph so that a binary image will be represented by several graphs, each of which consists of a number of segments. Two segments are called connected, if they have at least one node in common or if there is a sequence of connected segments.

The classification of line elements and nodes is similar to approaches described by Sakai et al. [1] and Landy and Cohen [2] and is based on the 8-neighbourhood.

2.1 Pixel classification

For the classification we need two characteristics of a pixel P. N(P) is the number of black 8-neighbours of P. T(P) is the number of black to white, and white to black changes in the 8-neighbourhood of P.

Based on these definitions we can classify each pixel P as follows:

a) $N(P) = 0$ → P isolated point
b) $N(P) \leq 2$ and $T(P) = 2$ → P end point
c) $N(P) = 2$ and $T(P) = 4$ → P line element
d) $N(P) > 2$ and $T(P) = 4$ → P node candidate
e) $N(P)$ arbitrary $T(P) > 4$ → P X- or T-node

In the case of d) there will be at least two adjacent nodes that are represented by only one node in the symbolic description. In this case there is no bijective mapping of the symbolic nodes to their candidates. To solve this problem we take the first node candidate as a node and then we look for line elements among the more distant neighbours of P. Thus we get a connection from the line elements to node P, and the corresponding connection points in the 8-neighbourhood of P, which could be node candidates, will be defined as line elements. A flow diagram of the implemented algorithm is shown in Fig. 2. The algorithm starts at a node. Thus we have a starting loop which finds one of the three different types of nodes. Determining a segment requires the following operations: First, the beginning node has to be found, then the line elements are determined, and finally the end node is found. Using the node stack, the algorithm checks whether there are nodes and line segments to be processed.

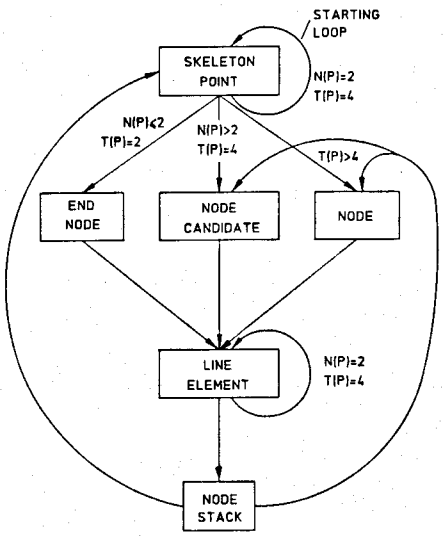

Figure 2
Flow diagram of the decomposition algorithm

2.2 Classification of line segments

The result of the pixel classification is the description of the binary image represented by the skeleton. The next step is the classification of individual line segments. A-priori knowledge is used to separate the graphic information into categories such as characters and the different types of lines that are used in mechanical construction drawings. For this purpose we use as features the thickness and number of segments in a graph. One type of lines are the border lines which are part of the graph with the largest number of segments and which are thicker than other lines. Usually due to noise, the thickness of a line is not constant. In particular, there will be disturbances at the ends of segments as for example at a free end of a line or in arrows. Therefore, characteristic values for the line thickness were calculated by averaging the thickness only over those segment points that are sufficiently far away from the segment end points. Another method is to replace the average by the median. The median method turned out to work better in experimental tests.

In order to define a quantitative criterion of line thickness we need a measure of thickness for each point of a line. Let P be a skeleton point of a line in a binary image. The line thickness of point P is defined as the maximum of the 8-neighborhood of P in the distance transform assuming 4-connectedness:

$$\text{Thickness}(P) = \max\{\text{dist}_4(N.8(P))\},$$

where N.8(P) are the points in the 8-neighbourhood of point P.

The method described above was tested on a number of mechanical construction drawings. The experimental result showed that about 5% of the segments were not correctly classified. These errors are expected to be reduced by exploiting the a-priori knowledge that border lines are connected.

So far we have considered the graph that is the largest in terms of the number of segments. The other graphs represent hidden border lines, construction lines, or alphanumeric characters or mathematical symbols. Usually, a graph representing a character consists of more than one and less than ten segments. Additional features for the purpose of recognition could be the preference directions and unconnected segments on imaginary lines.

3. CURVE APPROXIMATION

Each curve is approximated by a sequence of curve primitives the parameters of which have to be determined in such a way that a global criterion is optimized. This method is similar in spirit to those described in [3, 4], although there are differences in the global recognition criterion and in the search procedure that finds the best 'explanation' in terms of breakpoints and curve primitives.

3.1 Dynamic programming

For a given curve $C = (x_i)_{i=0,\ldots,I}$, the problem is to find among all possible sets of breakpoints and allowed curve primitives, which approximate the partial curves between these breakpoints, the one that minimizes the sum of certain partial costs describing the approximation quality on the partial curves between the breakpoints. Such a global optimization problem can be solved by dynamic programming. This technique divides the whole problem into a sequence of smaller ones that can be solved step by step.

The recursion formula is

$$A(0) := 0$$

$$(*)\quad A(i) := \min_{0 \leq j < i} \{A(j) + a(j,i)\} \quad (0 < i \leq I)$$

with the accumulated costs A and the additional costs a, the goal is to minimize A(I).

The basic assumption to be made is the optimality principle which in this case can be stated as follows: If an approximation of C with breakpoint indices $0=i_1,\ldots,i_k,\ldots,i_K=I$ and curve primitives $P_1,\ldots,P_K$ is optimal with respectto a criterion (*), then the approximation up to $i=i_k$ is optimal for the partial curve up to x_{i_k}.

In our approach, the additional costs consist of a term describing the quality of the approximation and a term depending on the (type of) curve primitive:

$$a(j,i) := \min\{d(P,C_{ji}) + p(P): P \in S_{ji} = \text{a certain set of curve primitives}\}.$$

$d(P,C_{ji})$ will generally be some distance measure between a curve primitive P (e.g., a line or circular arc) and the partial curve C_{ji} from x_j to x_i. The penalty p(P) is an extra cost for each curve primitive. As a

typical example, let for all j, i, S_{ji} be the set of all lines on the plane, d(L,Cji) be the sum of the squared euclidean distances of the points $x_j,\ldots,x_i$ to the line L, and p(L) be a constant c ≥ 0. Then (*) reduces to

$$A(i) := \min_{0 \leq j < i} \{A(j) + c + \min_{\text{lines } L} d(L,C_{ji})\} .$$

Minimizing A(I) means to find a line approximation of C, which minimizes both the sum of all deviation errors and the number of lines, the latter weigthed with c. A big penalty value c results in a coarse approximation by few lines and vice versa. The number K of curve primitives is not specified in advance but a result of the recognition process. For the total deviation error Δ of an approximation with l lines, the inequality Δ ≥ A(I) - l•c holds. In order to construct the optimal solution, it is suitable to keep track of the decisions taken in (*) in an additional backpointer array.

Generally, the dynamic programming approach described above finds the global optimum minimizing the final accumulated cost A(I). Its flexibility offers a wide range of variations:

The set of potential curve primitives S_{ji} may e.g. consist of straight lines, circular arcs, splines etc., or subsets of these, e.g. curves only, which pass through the breakpoints x_j, x_i, only lines of specific directions, and so on. It is an important advantage that a-priori knowledge can often very easily be incorporated in the recognition algorithm.

The deviation measure d may be a distance measure or more generally any cost function that expresses the quality of the approximation. If, e.g., the distance between a curve primitive and a partial curve should be smaller than a value R, define

$$d(P,C_{ji}) := \begin{cases} \text{dist}(P,C_{ji}) & \text{if } \text{dist}(P,C_{ji}) < R \\ \infty & \text{else .} \end{cases}$$

The penalty p counterbalances the algorithm's tendency of producing a vast number of breakpoints in order to find an approximation with the least deviation possible. It also helps counterbalancing with respect to different subsets of S_{ji}: Imagine, for example, S_{ji} = {lines} ∪ {circles}. Since a line can be viewed as a circle with infinite radius, typically for a digital curve a better approximation can be reached by a circle rather than a line. This effect can be compensated by the constraint: p(circle) > p(line).

3.2 Pruning

The computational complexity of the dynamic programming algorithm can be rather high. In order to reduce the search effort, it is useful to investigate pruning strategies which reduce the search effort significantly: Instead of the i points j = 0,...,i-1 in formula (*), most of which are unlikely candidates forthe best fitting predecessor of i, we only investigate indices j ∈ M(i), where M(i) is a suitably chosen non-empty subset of {0,...,i-1}. The following two pruning variants have been developed:

1. Let M(i) be those predecessors j of i, the accumulated score A(j) of which is not too bad with respect to A(i-1), i.e. choose a suitable real number δ of about the size of p and define

M(1) := {0}

M(i) := {j ∈ M(i-1) ∪ {i-1}: A(j) < A(i-1) + δ} (0 <i ≤I).

If the pruning value is smaller than the penalty for a new curve primitive, too many good candidates directly preceeding i are pruned. So in that case, choose a suitable integer k and modify the definition according to

M(i) := {j ∈ M(i-1): A(j) < A(i-1) + δ} ∪ {i-k,...,i-1}.

2. Points on a straight line segment are unlikely to be breakpoints and should be pruned. Choose a sufficiently small value ε and an integer k and define

$M_{k,\varepsilon}(i)$:= {j: either (0 ≤ j ≤ k or i-k < j ≤ I), or there is a line that approximates the partial curve $C_{j-k,j+k}$ better than ε} .

Clearly, these pruning strategies can be combined and extended. Only method 1, with parameters derived from a few tests, reduced the search space by a factor of 10 without significant effect on the approximation accuracy.

4. EXPERIMENTAL RESULTS

Experimental tests were performed for straight line segments as curve primitives. The error criterion of each segment was the sum over the squared distances between the straight line and the curve points. Computing the best fitting straight line for a given curve segment then amounts to determining the smallest eigenvalue of a two-dimensional scatter matrix. By using the method of running sums, the overall cost for computing $d(P,C_{ji})$ is independent of the number of points of the curve segments.

Therefore, the computational complexity of the full search algorithm is proportional to the square of overall number of curve points. A similar time complexity was obtained for a dynamic programming approach to nonlinear smoothing [5]. The important feature of the approach presented here is that the global optimality of the solution is guaranteed, which is not true for a number of other approaches, such as the Split-and-Merge-Algorithm for polygonal approximations by Pavlidis [pp. 179, 6].

Fig. 3 shows the application of the algorithm to a freehand drawing for varying degrees of approximation. The points of the approximation corresponding to breakpoints are marked by

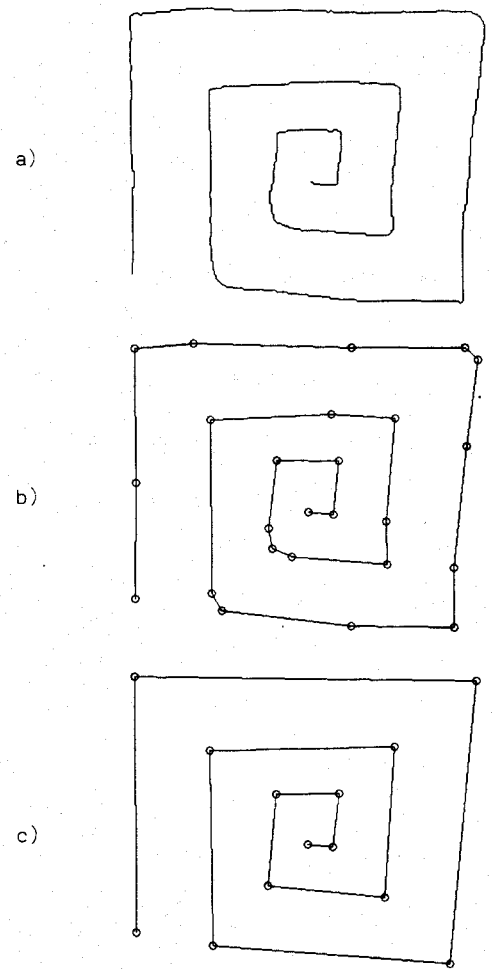

Figure 3
Application of curve fitting to a freehand drawing: a) freehand drawing, b), c) results of curve fitting for two degrees of approximation

circles. The deviations of the straight lines from the original picture are negligible in Fig. c, where appropriate penalties are chosen. Fig. 4 shows the resulting straight-line approximations for a mechanical construction drawing. In this case the distortion is visible for curved lines only.

5. SUMMARY

We have studied the application of optimal curve fitting to skeletized images for the purpose of pattern recognition. The optimization is based on dynamic programming, the computational complexity of which can be drastically reduced by pruning unlikely hypotheses. Experimental tests have been presented for freehand drawings and mechanical construction drawings.

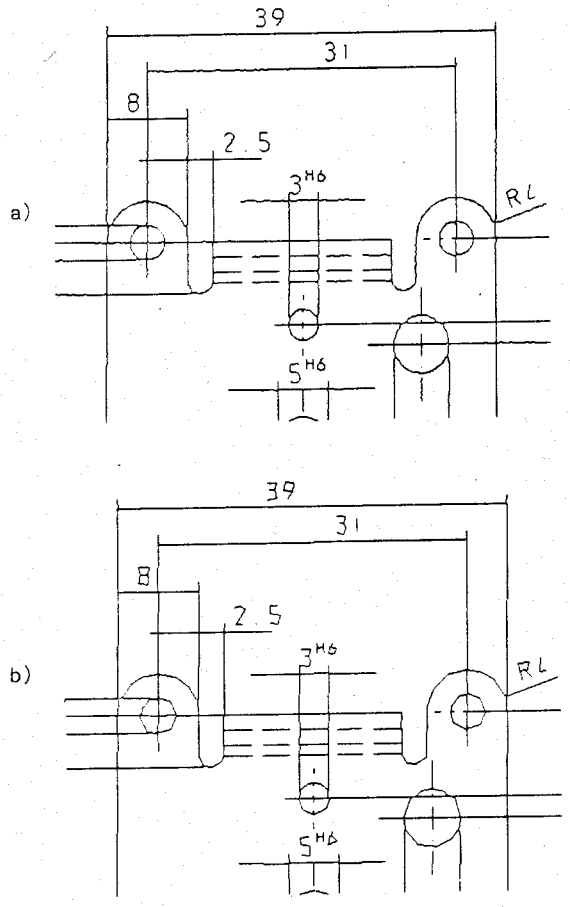

Figure 4
Application of curve fitting to a mechanical construction drawing: a) skeletized input image, b) result of curve fitting

REFERENCES

[1] T. Sakai, M. Nagao and H. Matsushima, Extraction of Invariant Picture Sub-Structures by Computer, in: Computer Graphics and Image Processing 1, (1972), pp. 81-96.

[2] M.S. Landy and Y. Cohen, Vectorgraph Coding: Efficient Coding of Line Drawings, in: Computer Vision, Graphics and Image Processing 30, (1985), pp. 331-344.

[3] G. Papakonstantinou, Optimal Polygonal Approximation of Digital Curves, in: Signal Processing, vol. 8 (1985), pp. 131-135.

[4] J. Mason, Contour Recognition: A Syntax-Directed Approach, IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Tokyo, 39.4.1-4, 1986.

[5] H. Ney, A Dynamic Programming Algorithm for Nonlinear Smoothing, in: Signal Processing, vol. 5, pp. 163-173.

[6] T. Pavlidis, Structural Pattern Recognition, Springer-Verlag, 2nd printing, New York, 1980.

SIGNAL PROCESSING IV: THEORIES AND APPLICATIONS

Proceedings of EUSIPCO-88
Fourth European Signal Processing Conference
Grenoble, France, September 5–8, 1988

Edited by

J. L. LACOUME
National Polytechnic Institute of Grenoble
Grenoble, France

A. CHEHIKIAN
Joseph Fourier University
Grenoble, France

N. MARTIN
National Center for Scientific Research (CNRS)
France

J. MALBOS
National Center for Scientific Research (CNRS)
France

VOLUME III

1988

NORTH-HOLLAND – AMSTERDAM • NEW YORK • OXFORD • TOKYO