Fast Guaranteed Polygonal Approximations of Closed Digital Curves

Fabien Feschet

LLAIC1 Laboratory
IUT Clermont-Ferrand, Campus des Cézeaux63172 Aubière Cedex, France
feschet@llaic3.u-clermont1.fr

Abstract. We present in this paper a new non-parametric method for polygonal approximations of digital curves. In classical polygonal approximation algorithms, a starting point is randomly chosen on the curve and heuristics are used to ensure its effectiveness. We propose to use a new canonical representation of digital curves where no point is privileged. We restrict the class of approximation polygons to the class of digital polygonalizations of the curve. We describe the first algorithm which computes the polygon with minimal Integral Summed Squared Error in the class in both linear time and space, which is optimal, independently of any starting point.

1 Introduction

Polygonal approximation of digital curves or shapes is an important task in pattern recognition, digital cartography, data compression... Polygonal models are still easier to handle than digital curves. The problem is usually defined by an error criterion and the constraint that the vertices of the polygonal model are points of the original digital curve. In the present work, we only use the Integral Summed Squared Error (ISSE) criterion. We refer to the thesis of Marji [Mar03] which contains more than 100 algorithms, methods and measures. Polygonal approximations are closely related to dominant [TC89] and corner [MS04] point detection. The polygonal approximation problem consists in finding the real polygon such that the ISSE with the digital curve is minimized. This problem is ill-posed since the polygon obtained by linking every point of the digital curve clearly reaches the minimal value of 0. This fact leads to the $\min -\varepsilon$ problem which corresponds to the minimization of the ISSE for polygons with a fixed number of edges, or to the non-parametric algorithms where the number of edges is automatically defined. Two classes of algorithms have been proposed: graph-theoretical [II88] and optimization (dynamic programming) [PV94]. Best complexities are $O(n^2 \log n)$ for the former and closed to $O(n^2)$ time and $O(Mn)$ space where M is the fixed number of edges [Sal01, Sal02] for the latter and n is the number of points. Beside this, suboptimal methods have also been proposed [DP73, RR92, MS04, KF04]. They have the advantage of having low complexity, but they lack guarantee for high quality results.

H. Kalviainen et al. (Eds.): SCIA 2005, LNCS 3540, pp. 910–919, 2005.

In almost all previous works, digital curves were considered opened and so for closed curves an initial starting point was randomly chosen. Heuristics have been proposed to overcome the problem [PV94, HL02]. We propose in this paper a non-parametric approach to the problem by combining graph-theoretic and optimization approaches in order to solve the initial point problem. The result is an algorithm which does not depend on any point of the digital curve. Our algorithm is based on digital lines - partially used by Cornic [Cor97] -, a circular arc-graph canonical representation of digital curves [Fes05] and efficient error computation following the ideas in [HL02]. The resulting algorithm is linear time and linear space and is thus optimal. Starting with digital lines, we construct a graph representation. To obtain an efficient algorithm, we restrict the class of allowed polygonal models to the class of digital polygonalizations. We solve the polygonal approximation problem using the graph structure. Our strategy does not suffer from the tendency of the ISSE to favor polygonal model with a high number of edges since it is known [FT03] that there exists only two lengths differing by only one for the whole set of polygonalizations of any digital curve.

In Section 2, we present the notion of digital lines and the graph representation of digital curves known as the tangential cover. The method is presented in Section 3 and experimental validation in Section 4. Final conclusions and extensions end the paper.

2 Representation of Closed Digital Curves

A closed digital curve C is a list of connected points $p_i = (x_i, y_i)$ of $\mathbb{Z}^2$. The allowed connectivities are the standard four and eight connectivities corresponding to the points with L_1 and respectively L_∞ distance of one of a given point. Self-intersections are allowed as soon as points are duplicated in the list. For closed digital shapes, any boundary tracking algorithm produces the required list and the choice of the point p_0 is arbitrary. If $C = (p_i)_{0 \leq i < n}$, we call n the length of the curve. The curve is closed if $p_n = p_0$. The order given by the usual order of the indices defines the orientation of the curve. In the paper, left and right as well as next and previous are defined relatively to the orientation of C. Indices are intended modulo n. A polygonal approximation of the curve C is a real polygon whose vertices belongs to C. It has the same orientation than the curve C. The goal of this section is to construct a geometrical representation suitable for computing polygonal approximations.

2.1 Digital Lines

Our representation has been introduced in [FT99], used in [FT03] and fully exploited in [Fes05]. It is based on digital lines (we refer to Rosenfeld and Klette [RK04] for definitions and main properties of digital lines). A digital segment is a finite connected subset of a digital line. Given a list of points extracted from a digital curve C, there exist algorithms to incrementally recognize if the list is a digital segment or not. When the answer is positive, the algorithms output the

parameters of a digital line containing the digital segment. The points can be taken both in positive or negative orientation of the curve C. The complexity of the recognition is linear in the number of points of the list. Given a subset of C which is a digital segment, it is called maximal if and only if it cannot be extended over C to another digital segment. Maximality of digital segments can be checked during their recognition within the same complexity bound.

2.2 Tangential Cover

We now briefly present the construction given in [Fes05]. It is based on the notion of discrete (or digital) tangent at a point p_i of C. A digital tangent is a subset $(p_{i-l}, \ldots, p_i, \ldots, p_{i+r})$ which is a maximal segment. The construction is as follows: points are alternatively added to the right and to the left of p_i while the resulting subset is a digital segment. When one side cannot be further extended, the addition of points are pursued at the other side while the subset is a digital segment. The resulting set forms the digital tangent of C at p_i and is as symmetric as possible. However, symmetry is not imposed and l and r are usually different. Any digital tangent is a maximal digital segment. We denote by T_{p_i} the digital tangent at p_i.

As explained in [Fes05], it often happens that $T_{p_i} = T_{p_{i+1}}$. This property was exploited in [FT99, Fes05] to built an efficient algorithm to construct the set of all digital tangents $\mathcal{T}(C) = \{T_{p_i},\ p_i \in C\}$. The set $\mathcal{T}(C)$ is called the tangential cover of C. The complexity of its construction is proved to be $O(n)$. The tangential cover has the property that is contains all maximal digital segments which can be constructed on the curve C with connected subsets.

Each digital tangent can be represented by its left and right ending points p_{i-l} and p_{i+r} and by the parameters of the associated digital segment. It is straightforward to see that the series of l and r are increasing series and that two consecutive tangents must overlap. Moreover due to the maximality of the discrete segments, no tangent can be contained in another one. To represent the tangential cover, we use the concept of circular arc graphs [Sch03]. A digital tangent can be represented by an interval $[i - l, i + r]$. We associate to it the arc between the angles $2(i - l)\pi/n$ and $2(i + r)\pi/n$. We obtain a circular arc graph (see Fig. 1, right). We have increased or decreased the radius of each arc to distinguish between overlapping arcs. It must be said that each point can be viewed has a radial line from the center of the representation to one point of the boundary of the central unit circle. All the intersected arcs by this line correspond to the digital tangents containing the point. The horizontal axis pointing to the right corresponds to the points $(0, 0)$ of the left image. It is clear from the previous embedding that the tangential cover is canonical meaning that the arbitrary starting point does not influence the resulting graph.

We put on the graph the orientation of the curve C. For a tangent T_{p_i}, there exists a finite subset of tangents T_{p_j} such that T_{p_i} and T_{p_j} overlap and i is at the left of j. We construct the function $F(.)$ by mapping T_{p_i} to the overlapping tangent T_{p_j} with maximal j intended modulo n, that is following the orientation of the graph. A digital polygonalization of C is a succession of digital segments

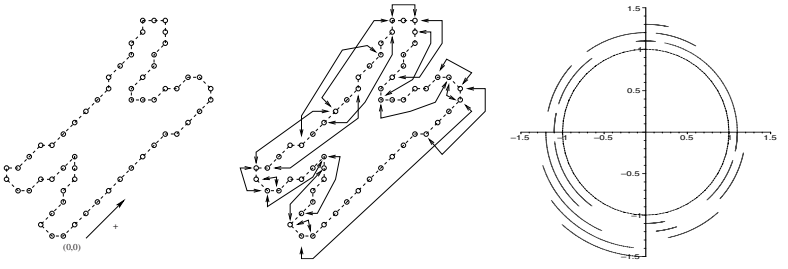

Fig. 1. (left) the chromosome shape (middle) maximal digital segments (right) its tangential cover

covering C, sharing only their ending points - the ending point of a digital segment is the beginning point of the next segment - and such that all but the last segment are maximal. This concept is well used in shape description [ZL04] or shape evolution [LL99]. Polygonalizations can be deduced from the function $F(.)$ [FT03],

Property 2.1. *For any tangent T in $\mathcal{T}(C)$, the digital segment starting at the right limiting point of T and ending at the right limiting point of $F(T)$ is a maximal digital segment.*

Thus using iterates of the function $F(.)$ and the maximality of any tangent, it is possible to build any polygonalizations of the curve. For instance on the chromosome shape of Fig. 1 containing 60 points, points from 59 to 13 forms a maximal digital segment. The maximal segment starting at 59 ends at 13 and the maximal digital segment starting at 13 ends at 15 and so on.

We now introduce the tangent $F^*(T)$ for any tangent T defined as the tangent such that: $F^*(T) = F^j(T)$ for some j and there exists a digital tangent in $\mathcal{T}(C)$ containing both the ending point of $F^*(T)$ and the beginning point of T. In other words, $F^*(T)$ is the last maximal digital segment of the polygonalization starting at the beginning point of T.

3 Polygonal Approximations

3.1 Context

The class of allowed polygonal approximations is exactly the class of digital polygonalizations of the curve C from any of the starting point of the digital tangents of the tangential cover $\mathcal{T}(C)$. We believe that this class is sufficiently rich. Moreover all other points of C appear inside straight parts and do not appear to be dominant or corner points.

To measure the error between a polygonal approximation and the original curve C, we use the classical ISSE criterion. We consider two points $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ of C. We introduce $a = y_j - y_i$, $b = x_j - x_i$ and $c = x_j y_i - y_j x_i$

such that all points (x, y) of the real line passing through p_i and p_j satisfies $ax - by + c = 0$. Each point p_k of C with $i \leq k \leq j$ are projected onto this line using perpendicular projection and the resulting distance $d_{ij}(p_k)$ is $d_{ij}^2(p_k) = (ax_k - by_k + c)^2 / \left(a^2 + b^2\right)$. The ISSE value between p_i and p_j is $\text{ISSE}(p_i \rightarrow p_j) = \sum_k d_{ij}^2(p_k)$ such that

$$\begin{aligned}\left(a^2 + b^2\right) \text{ISE}(p_i \rightarrow p_j) =& a^2 \sum_k x_k^2 + b^2 \sum_k y_k^2 + c^2 \sum_k 1 \\ &+ 2ac \sum_k x_k - 2bc \sum_k y_k - 2ab \sum_k x_k y_k\end{aligned} \tag{1}$$

If the sums of eq. (1) can be computed in linear time then so is the ISE criterion. The ISSE of a complete polygonal approximation is defined as the sum of the ISE for each segment of the real polygon.

3.2 Method

To compute a polygonal approximation of C, we start by computing the tangential cover $\mathcal{T}(C)$. We call $\mathcal{P}(C)$ the class of all allowed polygonal approximations. Our goal is to compute the element P of $\mathcal{P}(C)$ such that $\text{ISSE}(P) = \min\{\text{ISSE}(Q),\ Q \in \mathcal{P}(C)\}$.

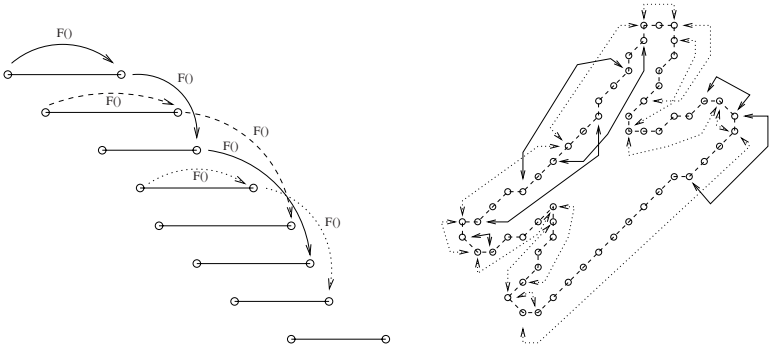

Fig. 2. (left) The iterates of $F(.)$ (right) cycles and paths for the chromosome shape

The first step of the algorithm consists in computing the function $F(.)$ (the algorithm is described in the next subsection). Having the function $F(.)$, we now consider a polygonalization, depicted in Fig. 2 (left). To compute the ISE value of a polygonalization, we must compute two partial ISE values: the first one is the ISE of each tangent and the second one is the ISE between a tangent T and the tangent $F(T)$. Those two computations are done in the second step of the method. Note that in fact we compute the six sums of equation (1).

The third step the method is a decomposition of the graph of the tangential cover into cycles and paths. To introduce those notions, let us fix an initial

tangent for instance $T_0 = T_{p_0}$. If we consider the tangents $T_j = F^j(T_0)$, then the list is obviously finite. Hence, there exists a minimal iterates j_0 such that $F^{j_0+1}(T_0) = F^k(T_0)$ with $k < j_0 + 1$. The set of tangents from T_0 until T_{j_0} forms a cycle in the graph. Let us consider now a tangent T' not belonging to the previous cycle. Then either the iterates $F^j(T')$ create a new cycle or they merge with a previously computed cycle and the portion of the graph from T' to the cycle is a path in the graph. For each path, we compute the label of the cycle it merges with. Those information are stored in each tangent. For the chromosome shape (see Fig. 2 right), there exists only one cycle, depicted in dotted lines, and paths merging to this cycle. The merging process implies that the end of the polygonalization starting at one tangent in a path must end with a tangent of the cycle. So, the function $F^*()$ is defined on a cycle and also concerns all tangents belonging to paths merging with the cycle.

Hence the fourth step consists in computing the $F^*()$ values. To do this, we rely on the merging labels previously computed. We first pick a tangent in a cycle and compute its $F^*()$ value using iterates of $F()$. Then, we move along the tangential cover graph and update $F^*()$ simply by checking if there still exists one tangent overlapping the current tested tangent and the current $F^*()$ tangent. So in one pass of the tangential cover graph, to each tangent is associated an $F^*()$ tangent.

To complete the computation of ISSE, we must end each polygonalization by the portion of the digital segment linking each tangent T and $F^*(T)$. This is the fifth step of the algorithm.How to do all steps in linear time is explained in next subsection.

3.3 Intermediate Constructions and Complexity

In first step, we compute $F()$. We start with $T = T_{p_0}$. By moving to the right, we determine $F(T)$ simply by checking the overlappings. This step obviously can be done in linear time with one complete turn over the tangential cover graph.

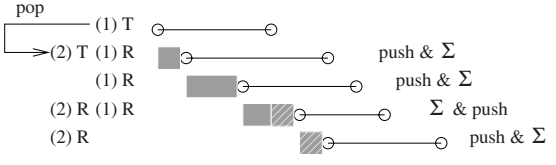

Fig. 3. ISE computation of each tangent

The computation process of the ISE from the beginning point of a tangent to its ending point is given in Fig. 3. T is the tangent being computed and R is a moving tangent. The numbers in parentheses mark which R are associated to which T. We use a FIFO (First-In First-Out) list. The sum symbol means summation in a global counter. The algorithm proceed by summing from the

beginning of T to the beginning of R (excluded) and push each portion in the list. When R reaches the next tangent of $F(T)$, the accumulation buffer contains the value of the six sums of the ISE computation. Then, we do not push since the computation is only partial at this step but we move T and pop from the list. The pop values are subtracted to the accumulation buffer. The process goes on but summation and push are inverted to push a complete part in the list. By moving all along the tangential cover graph, all partial ISE are computed, since we use the same strategy for the link between T and $F(T)$. The complexity is $O(\#\mathcal{T}(C))$ where # denotes cardinal.

To compute cycles and paths, the algorithm works in two steps. First, we initialize the cycle mark to 1 and mark T_{p_0} with it. We also mark $F(T_{p_0})$. We now consider T to be the next tangent of T_{p_0}. If it is not marked, we increase the cycle mark by one, mark T and $F(T)$ with the mark. If it is already marked, we propagate its mark to $F(T)$ if this last one is not already marked. At each tangent when a merging is detected, we store it in a lookup table. After a complete turn over the tangential cover, each tangent has a mark and all merging have been stored in the lookup table. However, it might happen that a path merges with another path before merging with a cycle. Hence, the lookup table must be modified in order to have a cycle mark for each path. For instance, let us consider the following lookup table $[1, 1, 3, 3, 2, 4, 4]$ which means that path 7 (cell with index 7 in the array) merges with path 4 and path 4 with cycle 3. The two cycles are 1 and 3 since the numbers in these cells correspond to the indices of the cells. We start with path 7 and move in the lookup table until we reach a cycle then we mark each element with the mark of the cycle. We then proceed with another unmodified element in decreasing order. In one pass, we get the correct lookup table $[1, 1, 3, 3, 1, 3, 3]$. Then each time we consider a mark, we access to the lookup table to get the merging cycle. So step 3 is also done in linear time.

To compute the $F^*()$ values, we first consider T_{p_0} and find $F^*(T_{p_0})$ and store it in a lookup table. We then consider the tangent T next to T_{p_0} in the tangential cover. If it does not have a $F^*()$ tangent, we look in the lookup table if an $F^*()$ has already been computed for its cycle. If not, we compute it and store in the lookup table. If there exists a previously computed $F^*()$, we move it to find $F^*(T)$ and store it in the lookup table. After, one complete turn over the tangential cover, every tangent T has an $F^*(T)$ tangent. The complexity is $O(\#\mathcal{T}(C))$.

The last step of the algorithm is similar to the previous one. But summation must be done to complete the computation of the ISE of each polygonalization.

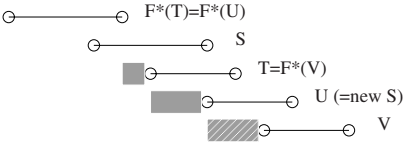

Fig. 4. ISE computation for the end of the polygonalizations

The process is depicted in Fig. 4. We suppose that T and $F^*(T)$ have been computed. To end the computation of the ISE measure, we sum the portion of the curve in grey level and store it in T. When going to the next tangent U, we discover that $F^*(U) = F^*(T)$ since S also covered U. Hence, we accumulate the summation in the buffer and store it in U. Next tangent is V which has a different $F^*()$. So we set the buffer to zero and do the summation. In one turn of the tangential cover using the lookup table for the currently computed $F^*()$, we can complete the computation of the ISSE measure for each polygonalization. To see that the whole complexity is $O(n)$, we must ensure that the number of time a point is computed is independant of n. This fact can be deduce from the following remark: when $F^*()$ change this means that the new S, equals to U in Fig. 4, must be strictly at the right of S such that when the buffer is set to zero, the new accumulation covers points not accumulated before. Hence, we obtain a linear time complexity.

The following theorem summarizes the previous complexity analysis.

Theorem 3.1. *The computation of the digital polygonalization starting at a beginning point of a digital tangent and with minimal ISSE can be done in $O(n)$ time and space complexity for any closed digital curve C of length n.*

4 Experiments

Experiments were performed on different shapes part of which is given on Fig. 5. The first three are taken from [TC89] and the last two from [MS04]. Results are given in Table 1. Several facts appear. First, the method performs well since the numbers of detected vertices are not too high and the ISSE values are relatively small. Second, the quality of the results of our strategy is very stable. Its non-parametric nature must be kept in mind when comparing the results with other methods. Moreover, the complexity of our method is linear time and space. Our method also controls the L_∞ error measure since it is based on digital segments. When using eight connected digital segments, we can guarantee that the L_∞ error is strictly below 1. But, thickness of digital lines might be increased. For instance, for the Bird shape, the use of four-connected digital lines leads to a solution in 20 vertices with an ISSE of 65.50. If we used the concept of blurred segments [DRFR05] which forms a fuzzy-like extension of digital segments, when modifying the *blurness* of digital segments we can reach a solution in only 19

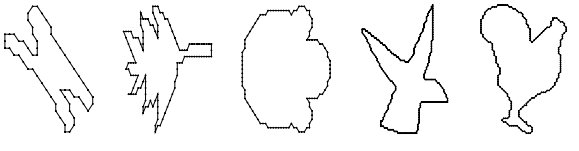

Fig. 5. Different shapes used for experimentation

Table 1. Results of experiments

Shapes	Methods	ISSE	# vertices	Shapes	Methods	ISSE	# vertices
Chromosome	Optimal	5.82	12	Semicircle	Optimal	14.40	15
	Optimal	3.13	17		Optimal	2.64	30
	[Cor97]	9.57	12		[Cor97]	13.00	22
	[TC89]	7.2	15		[TC89]	20.61	22
	[RR92]	4.81	18		[RR92]	11.5	27
	[SRS03]	8.53	17		[SRS03]	7.29	30
	New	5.90	14		*New*	14.59	21
Leaf	Optimal	22.42	17	Bird	[MS04]	72.92	15
	Optimal	6.80	28		*New*	34.93	36
	[Cor97]	25.80	23	Cock	[MS04]	39.96	24
	[TC89]	14.96	29		*New*	25.53	39
	[RR92]	14.18	32				
	[SRS03]	59.92	32				
	New	13.77	25				

vertices with an ISSE of 54.70. Our algorithm applies without any modifications since it is only based on the graph structure. Moreover, another extension can be used to also increase the quality of our method by allowing any polygonalizations of C. We currently cannot proved that our method remains linear time but experiments show computing times coherent with a linear time complexity.

5 Conclusions

We have proposed a new non-parametric algorithm for polygonal approximation of digital curves. It is based on digital lines and a canonical graph representation. By restricting the polygonal models to the digital polygonalizations starting at beginning points of digital tangents, our algorithm finds the polygonal model with the minimal ISSE value. The algorithm is linear in time and space complexity. A work is in progress to extend our representation to thick digital lines in order to increase the size of the class of polygonal models. We believe that we will nearly always find the global optimal polygonal model in linear time.

References

[Cor97] P. Cornic. Another look at the dominant point detection of digital curves. *Pattern Recognition Letters*, 18:13–25, 1997.

[DP73] D.H. Douglas and T.K. Pleucker. Algorithm for the reduction of the number of points required to represent a line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[DRFR05] I. Debled-Rennesson, F. Feschet, and J. Rouyer. Optimal blurred segments decomposition in linear time. In *Digital Geometry and Computer Imagery*, volume 3429 of *LNCS*, pages 371–382. Springer-Verlag, 2005.

[Fes05] F. Feschet. Canonical representations of discrete curves. *Pattern Analysis and Applications*, 2005. Accepted for publication, to appear.

[FT99] F. Feschet and L. Tougne. Optimal time computation of the tangent of a discrete curve: application to the curvature. In *DGCI*, volume 1568 of *LNCS*, pages 31–40. Springer-Verlag, 1999.

[FT03] F. Feschet. and L. Tougne. On the Min DSS Problem of Closed Discrete Curves. In A. Del Lungo, V. Di Gesù, and A. Kuba, editors, *IWCIA*, volume 12 of *ENDM*. Elsevier, 2003.

[HL02] J-H. Horng and J.T. Li. An automatic and efficient dynamic programming algorithm for polygonal approximation of digital curves. *Pattern Recognition Letters*, 23:171–182, 2002.

[II88] H. Imai and M. Iri. Polygonal approximations of a curve (formulations and algorithms). In G.T. Toussaint, editor, *Computational Morphology*, pages 71–86. North-Holland, 1988.

[KF04] A. Kolesnikov and P. Fränti. Reduced-search dynamic programming for approximation of polygonal curves. *Pattern Recognition Letters*, 24:2243–2254, 2004.

[LL99] L.J. Latecki and R. Lakämper. Convexity rule for shape decomposition based on discrete contour evolution. *Computer Vision and Image Understanding*, 73(3):441–454, 1999.

[Mar03] M. Marji. *On the detection of dominant points on digital planar curves.* PhD thesis, Graduate School, Wayne State University, 2003.

[MS04] M. Marji and P. Siy. Polygonal representation of digital planar curves through dominant point detection - a nonparametric algorithm. *Pattern Recognition*, 37(11):2113–2130, 2004.

[PV94] J.-C. Perez and E. Vidal. Optimum polygonal approximation of digitized curves. *Pattern Recognition Letters*, 15:743–750, 1994.

[RK04] A. Rosenfeld and R. Klette. Digital Straightness – a review. *Discrete Applied Math.*, 139(1–3):197–230, 2004.

[RR92] B.K. Ray and K.S. Ray. An algorithm for detecting dominant points and polygonal approximation of digitized curves. *Pattern Recognition Letters*, 13:849–856, 1992.

[Sal01] M. Salotti. An efficient algorithm for the optimal polygonal approximation of digitized curves. *Pattern Recognition Letters*, 22:215–221, 2001.

[Sal02] M. Salotti. Optimal polygonal approximation of digitized curves using the sum of square deviations criterion. *Pattern Recognition*, 35:435–443, 2002.

[Sch03] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency.* Berlin: Springer-Verlag, 2003.

[SRS03] B. Sarkar, S. Roy, and D. Sarkar. Hierarchical representation of digitized curves through dominant point detection. *Pattern Recognition Letters*, 24:2869–2882, 2003.

[TC89] C.-H. Teh and R.T. Chin. On the detection of dominant points on digital curves. *IEEE Trans. Pattern Anal. and Machine Intell.*, 11(8):859–872, 1989.

[ZL04] D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37(1–3):1–19, 2004.