A fast near-optimal *min*-# polygonal approximation of digitized curves

Alexander Kolesnikov[1] and Pasi Fränti[2]

[1] Institute of Automation and Electrometry
Pr.Ak.Koptyuga, 1, Novosibirsk, 630090, Russia
Email: kolesnikov@iae.nsk.su

[2] Dept of Computer Science, University of Joensuu,
BOX 111, FIN 80101, Joensuu, Finland
Email: franti@cs.joensuu.fi

ABSTRACT

We propose a fast near-optimal algorithm for solving the problem of *min*-# polygonal approximation of digitized curves. The algorithm consists of two steps. It first finds a reference approximation with minimum number of segments for a given error tolerance by using L_∞ error metrics. It then improves the quality of the approximation by a reduced-search dynamic programming with additive L_2 error measure. The algorithm is tailored for high-quality vectorization of digitized curves.

Keywords: vectorization, polygonal approximation, min-# problem, shortest path, dynamic programming

1. INTRODUCTION

We consider the problem of polygonal approximation of open digitized curves for high-quality vectorization tasks. The task is defined as optimal polygonal approximation of *N* vertices with the minimum number of linear segments *M* that satisfies a given error tolerance. It is known as the *min-# problem*.

The problem is closely related to the *min-$\in$ problem,* which aims at minimizing the approximation error for a given number of segments *M*. This problem can be solved by graph theory methods as proposed in $O(N^2\log N)$ time [1]. The problem can also be solved by dynamic programming algorithm in $O(N^2 M)$ time as proposed in [2]. Salotti has improved this approach by a method that works in $O(N^2)$ time [3]. Schuster and Katsagellos [4] have proposed another optimal algorithm with the time complexity of $O(N^2)$ based on the Lagrange multiplier method.

All the above algorithms are optimal but they have quadratic or cubic time complexity, which makes them impractical for large number of vertices *N*. In a recent paper [5], we have introduced fast near-optimal algorithm for the *min-$\in$ problem* that has time complexity remarkably less than $O(N^2)$.

Several algorithms for the *min-# problem* also exist. Graph theory method has been proposed in [6], and the complexity of this algorithm was then reduced to $O(N^2)$ in [1,4,7]. The dynamic programming approach also can be used to solve the problem, but complexity of the algorithm is $O(N^3)$ [2, 8].

A fast near-optimal algorithm was proposed for the *min-# problem* in [9]. The algorithm provides solution with minimum number of segments *M* for a given error tolerance d_T: $d \le d_T$. The approximation error *d* is defined as the maximum Euclidean distance from the vertices to the approximating segments, and it is the so-called L_∞ error metrics. The algorithm has been tailored especially to polygons with low number of segments, which is suitable as shape signatures in image retrieval from multimedia databases.

Technically, the algorithm in [9] can also be used for polygonal approximation in the vectorization tasks. However, the error metrics L_∞ is inferior to L_2 if we are dealing with approximation with low error tolerance for high-quality vectorization task. The L_2 error metrics (corresponding to the means squared error *E*) has also been considered in [9] but only as local distortion measure, and not as global cost function for the whole curve.

Thus, it is expected that the additive error metrics L_2 provides better results for the same number of approximating segments *M* in the vectorization application, but the error measure *E* can hardly be used as the error tolerance measure in the *min-# problem* because of its additive characteristic.

To solve this dilemma, we propose to use the L_∞ metrics as the input control parameter d_T, and the additive error measure *E* with metrics L_2 as the cost function in the optimization. In polygonal approximation of the lines in engineering drawings, maps, schemes, etc., the distortion tolerance d_T can be set up to half of the line width [10], or to 1-2 pixels for polygonal approximation of region borders in segmentation tasks.

In this work, we generalize the near-optimal algorithm solving the *min-$\in$ problem* [5] to solve the *min-# problem*, too. We formulate the *min-# problem* in two forms: *strong* and *weak*. The strong form means that the optimal solution with error metrics L_2 has to satisfy the constraint of the maximum distortion: $d \le d_T$. The weak form means that we are looking for an optimal solution that takes no account of the strong constraint on the distortion; it merely aims at finding solution for which $d \approx d_T$.

2. ALGORITHM IN THE WEAK FORM

Let us define an *open N-vertex polygonal curve* in 2-D space as the ordered set of $P_{1,N} = \{p_k: k = 1,...,N\}$. The approximating ($M$+1)-vertex polygonal curve is defined as $Q_{1,M+1}=\{q_k: q_k \in P, k = 1,..., M+1)$. The proposed algorithm consists of two steps:

> **Step 1:** *Minimize the number of segments M for a given constraint*: $d_1 \le d_T$.
>
> **Step 2:** *Optimize the reference solution with metrics L_2*

At the first step, we find a reference approximation with minimum number of segments M for a given error tolerance d_T using the algorithm proposed in [9] (algorithm A1). At the second step, we improve the quality of the reference approximation using a fast near-optimal algorithm with the cost function E (algorithm A2). The algorithm A2 is based on the reduced-search dynamic programming approach as proposed in [5].

2.1. Finding reference solution

We use the algorithm in [9] for generating the initial (reference) solution. It is based on the algorithm for finding single-source shortest path in directed acyclic graph (DAG) [4,7].

The algorithm A1 is represented in Fig. 1. The $R(j)$ gives the minimum number of segments in the polygon $Q_{0,j}$ connecting the start vertex p_0 and the current vertex p_j. The local distortion $d(i,j)$ is maximum Euclidean distance for the approximating segment (p_i,p_j).

To reduce the processing time, Schroeder and Laurent suggested to stop the further search when the current local distortion $d(i,j)$ is twice larger than the given error tolerance d_T. The B is an array of the parent vertices. The obtained solution defines the number of segments M and a reference solution.

```
Initialization
R(0) = 0

Recursion
FOR j = 1  TO  N  DO
   R(j)= ∞
   FOR i = j-i 0  DO
      IF(d(i,j) > 2 d_T)
         BREAK
      ENDIF
      IF(d(i,j) < d_T) AND (R(i) + 1 < R(j))
         R(j) = R(i) + 1, B(j) = i
      ENDIF
   END
END
```

Figure 1: Algorithm A1 [9] for the shortest path in the directed acyclic graph.

2.2. Optimize the reference solution

Let us define discrete *state space* Ω, where every point (n,m) in the space represents the sub-problem of approximating part of the input polygonal curve $P_{1,n}$ by m segments. Any output polygonal curve Q can be represented as a *path* in the state space Ω from the initial state (1,0) to the goal state (N,M). We also define $E(n, m)$ as the cost function of the optimal approximation for the state (n,m).

The solution obtained at the first step defines a reference path $G = \{(G(m),m): m = 0,…,M\}$ in the state space Ω. A bounding corridor is then constructed along the reference path G in the space (see Fig.3) that defines a bounded state space Ω_C. The left $\{L(m)\}$ and right $\{R(m)\}$ bounds of the corridor are defined as follows:

$$L(m) = \begin{cases} m+1; & m = 0,...,B_1, \\ \max\{\, m+1, G(m-B_1)\,\}; & m = B_1+1,...,M, \end{cases}$$

$$R(m) = \begin{cases} \min\{\, N, G(m+B_2)-1\,\}; & m = 0,...,M-B_2, \\ N; & m = M-B_2+1,...,M, \end{cases}$$

where $B_1 = \lfloor W/2 \rfloor$, and $B_2 = W - B_1$. The offsets $\{\Delta m(n)\}$ of the bounding corridor are defined relative to the bottom boundary of the state space Ω (see Fig.3):

$$\Delta m(n) = \begin{cases} 0, & n = 1,...,R(0); \\ \max\{0, m - B_1\}, & n = R(m-1)+1,..,R(m);\ m = 1,..,M. \end{cases}$$

Dynamic programming is performed inside the bounding corridor for solving the minimum cost function $E(n, m)$ using the recursive expression.

```
Initialization:
E(1,0) = 0

Recursion:
FOR m = 1  TO  M  DO
   FOR n = L(m)  TO  R(m)  DO
      C_min =  ∞
      FOR  j= L(m–1) TO R(m–1) DO
         C = E(j, (m–1) mod 2) + e^2(j, n)
         IF(C < C_min)
            C_min = C,     j_min = j
         ENDIF
      END
      E(n, m mod 2)    = C_min
      A(n, m – Δm(n)) = j_min
   END
END
E = E(N,M)

Backtracking to find the optimal path H(m):
H(M)= N
FOR m = M TO 1 DO
   H(m–1) = A(H(m), m – Δm(H(m)))
END
```

Figure 2: Algorithm A2 [5] for the reduced-search dynamic programming.

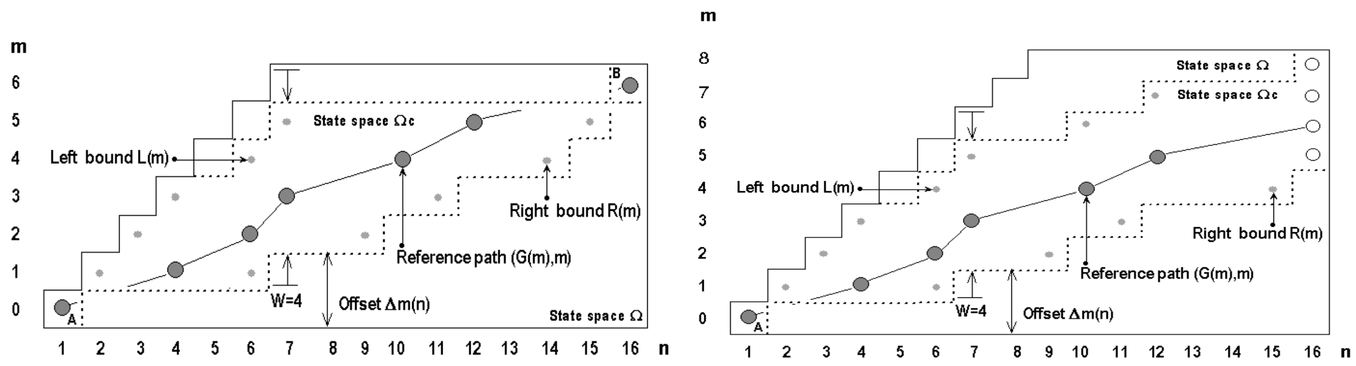

Figure 3. *Left*: The scheme of the bounding corridor for the problem size of N=16 and M = 6. The reference path G is marked with dark gray circles. The corridor width is W = 4, and the left (L) and right (R) bounds are marked with gray dots. *Right*: Scheme of the modified bounding corridor for the problem size of N=16. The corridor width is W = 4. The four final states are marked with circles, and the possible values of M are from 5 to 8.

The optimization algorithm A2 can be iterated using the output solution as a new starting point for next iteration. We can regulate the trade-off between quality and run time by changing the corridor width W, and the number of iterations. One iteration of the algorithm A2 is described in Fig. 2.

In practice, with two iterations for corridor width of W = 6-8 we can achieve 99-100% level of optimality. With the proper selection of the corridor width and number of iterations we can achieve the 100% level of optimality with high confidence [5].

2.3. Complexity of the algorithm

The processing time T_1 of the algorithm A1 for the error metrics L_∞ can be roughly estimated as $T_1 = O(N^2/M)$. In the case under consideration, polygonal approximation for precise vectorization, it means that the time complexity of the algorithm A1 ranges from $O(N)$ to $O(N^2)$ depending on the given error tolerance: smaller error tolerances would result in larger number of segments M and, thus, smaller time complexity. The space complexity of A1 is $O(N)$.

The processing time T_2 of the reduced-search algorithm A2 is proportional to N^2W^2/M per iteration. Thus, the time complexity of the algorithm A2 ranges from $O(N)$ to $O(N^2)$, too. This is also the time complexity of the whole algorithm (A1+A2). The exact time complexity depends on the given error tolerance, and on the parameter W (corridor width).

In the implementation, we use ring buffer of size $2\times N$ to store the values of $E(n,m)$. For the parent states A, we do not store the values in the absolute locations (n, m) but the relative locations $(n,m-\Delta m(n))$ defined by the offsets of the corridor boundary. In this way, we need an array of size $W\times N$ that covers the reduced state space Ω_C exactly Thus, the algorithm A2 needs space for $W\times N$ locations in the state space, and the corresponding memory requirements is $O(N)$.

3. ALGORITHM IN THE STRONG FORM

Solution for the weak form with fixed M can be found with the two steps of the represented algorithm. As for solution for the strong form, the situation is more complicated. We have to find a new number of segments M, which assure the constraint of the distortion measure $d\leq d_T$. It can be done in the following iterative way using bisection algorithm.

We first find a reference solution for the distortion tolerance d_T with the algorithm A1. We then optimize the reference solution with algorithm A2 and define the distortion d_2 of the optimized solution. If the distortion d_2 is greater than a given error tolerance d_T, we set a smaller error tolerance d^*_T (for example $d_T/2$). In this way, we define error tolerance range $[d^*_T, d_T]$ and apply algorithms A1 and A2 for the error tolerance d^*_T. We then repeat bisection of the tolerance range and the Steps 1 and 2 until proper solution is found. If the distortion d_2 is non-increasing function of the segments number M, the algorithm will converge.

For the optimal solutions the previous condition is usually fulfilled. To speed-up the convergence and to reduce the number of iterations we can use the search redundancy of the dynamic programming method. For this purpose we modified the bounding corridor to provide us with solutions for W final states (see Fig. 3). In this case, we have to use ring buffer of size $W\times N$ to store the values of $E(n,m)$ in the modified corridor. After every step of the optimization algorithm A2, we check the distortion of W optimal solutions with $m = M-B_2, \ldots, M+B_1-1$, and check whether we should continue the process, or if the final solution was already found. The number of iterations depends on the given error tolerance d_T, and on the corridor width W. The iterative process converges faster for larger error tolerance values than for smaller ones.

After that we can apply additional steps of the algorithm A2 to achieve nearly optimal result.

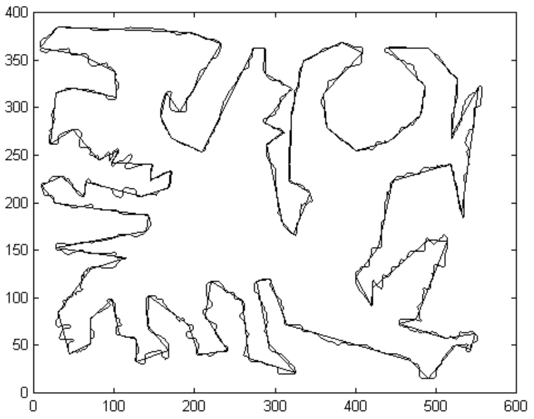
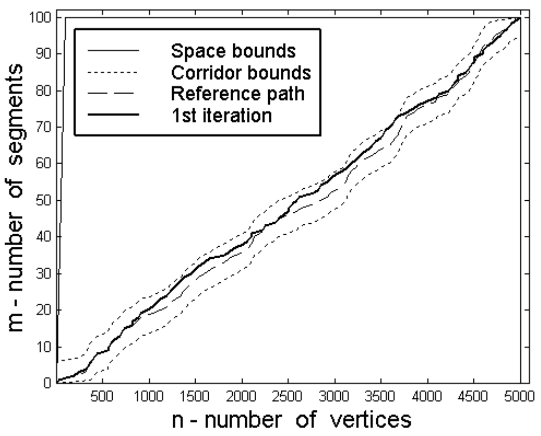

Figure 4. *Left:* approximation after the 1st iteration for a sample 5004-vertex (from [3]) with *M*=100 segments. *Right*: The bounding corridor in the state space with the width *W*=10.

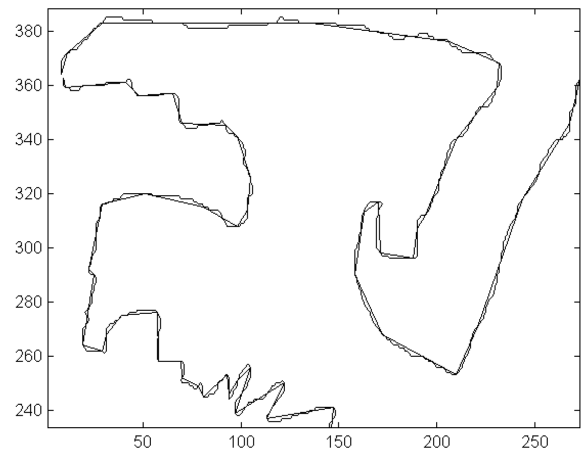
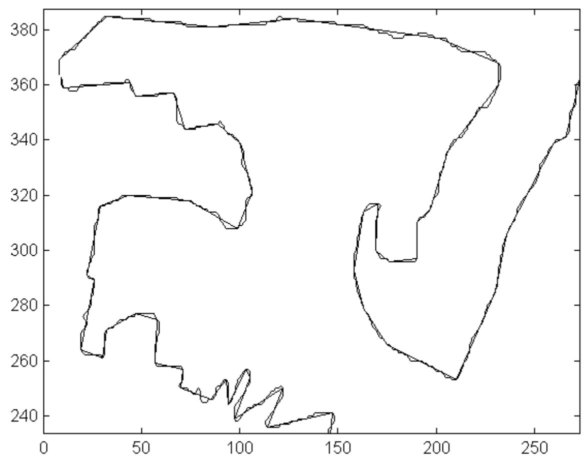

Figure 5: *Left*: Fragment of the 5004-vertices test shape (from [3]) after approximation with algorithm A1 for the error tolerance d_T=2 pixels, the number of segments *M* = 281, the approximation error E_1 = 4830. *Right*: Fragment of the 5004-vertices test shape after optimization of the reference solution with algorithm A2, the corridor width *W*=8, distortion *d*=2.55 pixels, number of segments *M* = 281, the approximation error E_2 = 2480.

4. RESULTS AND DISCUSSION

We concentrate here on the analysis of the algorithms A1 and A2 in the weak and strong forms. There are two questions to be answered: (a) do we gain from the use of the optimization algorithm A2, and (b) could we use the algorithm A2 in weak form instead of the strong one for polygonal approximation without increasing the number of segments in the vectorization task.

4.1. Efficiency of the algorithm in weak form

Let us compare the properties of the reference solution obtained by the algorithm A1, the result of the algorithm A2, and the optimal solution. The data for the 5004-vertices test shape are represented in Table 1. The run times have been obtained using Pentium II, 266 MHz processor.

Using the error metrics L_∞ for the optimization changes the characteristics of the approximation error distribution along the digitized curve. In the reference solution we can observe the appearance of long slits between linear segments and the digitized curve (see Fig.5). It can be explained that the width of the slits is smaller than the error tolerance.

The additive cost function *E* controls the global approximation error for the whole curve. Approximation error E_2 of the optimized solution is therefore twice as small as that of the reference solution. The approximating segments usually lie closer to the vertices of the curve with only a few exceptions. The maximum local distortion (d_2) of the solution A2 can be greater than the maximum local distortion (d_1) of the reference solution A1. However, this is the case only for a few vertices, about 1-2% of the total number of the vertices.

Table 1. Number of segments M, distortions (d_1 and d_2), mean squared errors (E_1 and E_2) and processing time (T_1 and T_2) of the algorithms A1 and A2 for the test shape (N = 5004).

d_T **(pixels):**	**0.5**	**1.0**	**1.5**	**2.0**	**2.5**	**3.0**
M**:**	932	508	344	281	234	199
d_1 **(pixels):**	0.50	1.00	1.50	2.00	2.50	3.00
d_2 **(pixels):**	0.80	1.80	2.50	2.60	3.74	5.00
E_1**:**	259	1158	2631	4832	7387	10685
E_2**:**	204	674	1473	2477	3838	5763
T_1 **(s):**	0.11	0.22	0.33	0.50	0.65	0.87
T_2**, (s):**	0.77	1.32	1.87	2.30	2.75	3.30
Total time:	0.88	1.54	2.20	2.80	3.40	4.17

4.2. Weak vs. Strong

The optimized solution for the problem in the strong form for the test shape can be achieved in 3-5 iterations, depending on the given error tolerance. Moreover, in the case of low error tolerance values, the solution for the strong form will have larger number of approximating segments: about 50% more than in the solution of the weak form.

Actually, in applied tasks such as vectorization, we usually define a space scale of the noise and treat it as characteristic value ($d \approx d_T$), not as a strong constraint $d \le d_T$. Therefore, we do not need to spent processing time to additional iterations of optimization to satisfy the strong condition $d_2 \le d_T$. In practice, we can use the solution with less number of segments that satisfies the weaker condition $d_2 \approx d_T$.

5. CONCLUSIONS

We have developed a fast near-optimal algorithm for the *min-# problem* of polygonal approximation for digitized curves. To combine the practicality of the metrics L_∞ and the high quality of the error metrics L_2, we proposed to use the distortion measure with metrics L_∞ as input control parameter d_T in polygonal approximation, and the additive error measure E with metrics L_2 as cost function for the optimization.

The *min-# problem* was considered in strong and weak forms. In the strong form, the solution with error metrics L_2 has to satisfy the constraint on the maximum distortion $d \le d_T$. The solution in the weak form takes no account of the constraint on the error tolerance of individual line segments but merely controls the algorithm by aiming at solution with $d \approx d_T$.

For the problem in the weak form, we proposed two-step algorithm: at the first step we find a reference approximation with minimum number of segments M for a given error tolerance d_T with error measure L_∞, and at the second step we improve the quality of the reference approximation using a fast near-optimal algorithm with the error metrics L_2.

Complexity of the algorithms varies between $O(N)$ and $O(N^2)$ depending on the error tolerance. The algorithm has the space complexity proportional to (NW).

For the problem in strong form, we constructed an iterative algorithm based on the two-step algorithm. To speed-up the convergence and to reduce the number of iterations, we introduced a modified bounding corridor of width W for reduced-search dynamic programming with controlled search redundancy that provides solutions for W final states.

The results showed that in practical application it is reasonable to use the algorithm in weak form for fast near-optimal approximation of digitized curves in applied tasks.

REFERENCES

[1] W.S.Chan, F.Chin, On approximation of polygonal curves with minimum number of line segments or minimum error, *Int. J on Computational Geometry and Applications*, *6*, 1996, 59-77.

[2] J.C.Perez, E.Vidal, Optimum polygonal appro-ximation of digitized curves, *Pattern Recognition Letters*, *15*, 1994, 743-750.

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

[4] G.M.Schuster, Katsaggelos, An optimal polygonal boundary encoding scheme in the rate distortion sense, *IEEE Trans. On Circuits and Systems for Video Technology*, *7*, 1998, 13-26.

[5] A.Kolesnikov, P.Fränti, A fast near-optimal algorithm for approximation of polygonal curves, *submitted to the International Conference on Pattern Recognition-IAPR'2002.*

[6] H.Imai, M.Irai, Polygonal approximation of a curve: formulations and algorithms, in *Computaional Moprhology* (G.T.Toussaint, ed., North-Holland, Netherlands, 1988, 71-86).

[7] G.M. Schuster and A.K.Katsaggelos, An optimal lossy segmentation encoding scheme, *Proc. Conf. on Visual Communications and Image Processing*, SPIE, Mar. 1996, 1050-1061.

[8] C.-C.Tseng, C.-J.Juan, H.-C.Chang, J.-F.Lin, An opti-mal line segment extraction algorithm for online Chinese character recognition using dynamic programming, *Pattern Recognition Letters*, *19*, 1998, 53-961.

[9] K.Schroeder, P.Laurent, Efficient polygon approximations for shape signatures, *Proc. Int. Conf. on Image Processing-ICIP'99*, *2*, 1999, 811-814.

[10] L.Wenyin, D.Dori, From raster to vectors: extracting visual information from line drawings, *Pattern Analysis & Applications*, *22*, 1999, 10-21.