Polygonal approximation of curves

www.kovalevsky.de, last update: 2010-10-26

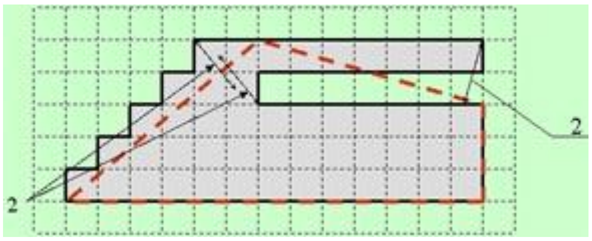
Let me know
what you think

The problem of polygonal approximation
Schlesingers similarity
The split-and-merge method
The sector method
Improved sector method
Optimal polygonal approximation

The problem of polygonal approximation

Given is a digital curve *C* in a 2D image. We want to approximate *C* by a polygon which has as few as possible edges and is "similar" to *C*. One of the possible criterions of the similarity is the Hausdorff distance: Let S_1 and S_2 be two sets of points. Let us denote the distance between the points *p* and *q* as *d*(*p*,*q*). Then the following definitions hold:

$D(p,S) = \min d(p,q) \forall q \in S;$	The distance from point *p* to set
$D(S_1,S_2) = \max D(p,S_2) \forall p \in S_1;$	The distance from set S_1 to set
$D(S_2,S_1) = \max D(q,S_1) \forall q \in S_2;$	The distance from set S_2 to set
$H(S_1,S_2) = \max(D(S_1,S_2), D(S_2,S_1)).$	The Hausdorff distance betwee



Sample: All three small arrows have the length = 2.

Here the Hausdorff distance between the black and the red curve is <2, however, the "true deviation" is much greater.

Schlesinger has suggested a more adequate measure of the similarity of curves.

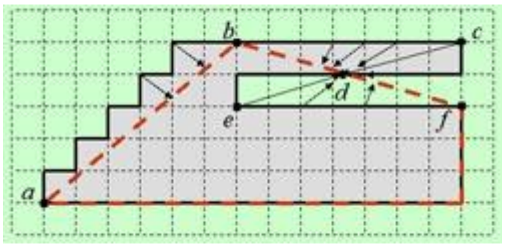
Schlesingers similarity

Given are two digital curves C_1 and C_2 . Take *m* points along C_1 and *n* points along C_2. Thus one gets two ordered sets M_1 and M_2 of points. The smaller is the distance between subsequent points the more precise the estimate of the similarity.

Let us denote the distance between two points *p* and *q* as $d(p, q)$.

Find the monotone map $F: M_1 \rightarrow M_2$, where monotone means that for any two points *p* and *q* of M_1 such that $p > q$ also $F(p) > F(q)$ holds. The sign " > " in "$p > q$" means that the point *p* follows the point *q* in the sequence M_1. We look for the smallest value of the maximum distance between the corresponding points. This value is the Schlesinger distance $DS(M_1, M_2) = \min(\max(d(p, F(p)))$;

Schlesinger has also suggested an efficient algorithm for computing the distance *DS* for any two digital curves. The value of *DS* for our example can be computed as follows. Let the black curve be *B* and the red one be *R*. They are subdivided into corresponding segments (*a*,*b*), (*b*,*d*), (*d*,*f*) and (*f*,*a*). Map each point of a segment of *B* onto the nearest point of the corresponding segment of *R* as shown in the figure.

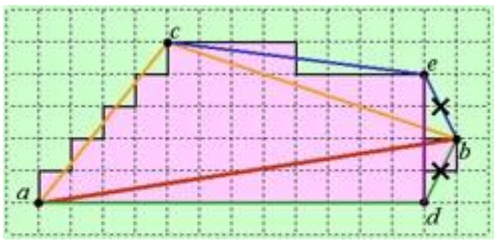


The value of *DS* for the above example is defined by the length of (*c*,*d*) (or (*e*,*d*)) and is equal to 3.6 that is essentially greater than 2.

Algorithms for the polygonal approximation: The split-and-merge method

The advantage of this method is that it is very simple to understand and to program. The idea is as follows. Given is the curve *C* and the maximum allowed distance ε from the points of *C* to the polygon. Take an arbitrary point P_1 of *C*. Find the point P_2 with the greatest distance from P_1. Now the curve is subdivided into two segments: (P_1, P_2) and (P_2, P_1). Each segment (P_i, P_k) has its chord (at start two segments have the same chord). For each segment find the point P_m with the greatest distance *d* from the corresponding chord. If $d>\varepsilon$ split the segment (P_i,P_k) into two segments (P_i,P_m) and (P_m,P_k) . Repeat this until $d\leq\varepsilon$ for all segments. The "split" phase is finished. Now check each pair of adjacent segments (P_i,P_m) and (P_m,P_k) whether the distance *d* of P_m to the chord (P_i,P_k) satisfies $d\leq\varepsilon$. If so, merge the segments (P_i, P_m) and (P_m, P_k).

An example with $\varepsilon = 1.5$



Split phase: $P_1=a$, $P_2=b$, $P_3=c$, $P_4=d$, $P_5=e$.

Merge phase: (e,b) and (b,d) have been merged to (e, d).

Algorithms for polygonal approximation: The sector method

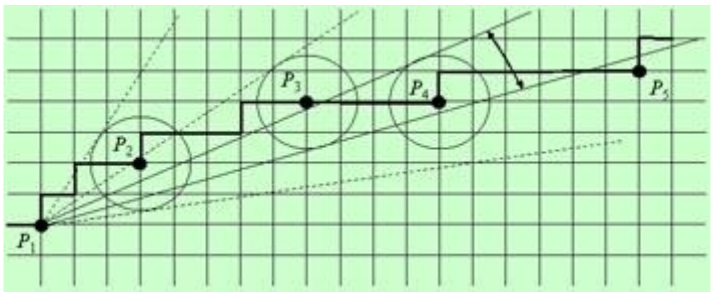
The split-and-merge method is rather slow because each point must be tested many times, depending upon the number of splitting of the segment to which it belongs. This method minimizes the Hausdorff distance, not the Schlesinger distance.

An efficient method called the sector method was suggested in the 70th by Williams. The idea is as follows:

1) Start with some point P_1 of the curve C. Test all subsequent points. For each point P_i with $i>1$ and $d(P_1, P_i)>\varepsilon$ draw a circle with the radius equal to the tolerance ε and draw two tangents from P_1 to the circle. The space between the tangents compose a sector, while each straight line L through P_1 lying in the sector has the property that the distance from the center of the circle to L is less than ε.

2) If the next point P_i (fig. P_3 or P_4) lies in the sector, then a new circle with its center at P_i and a sector S are constructed. The new sector is the intersection of S with the old one.

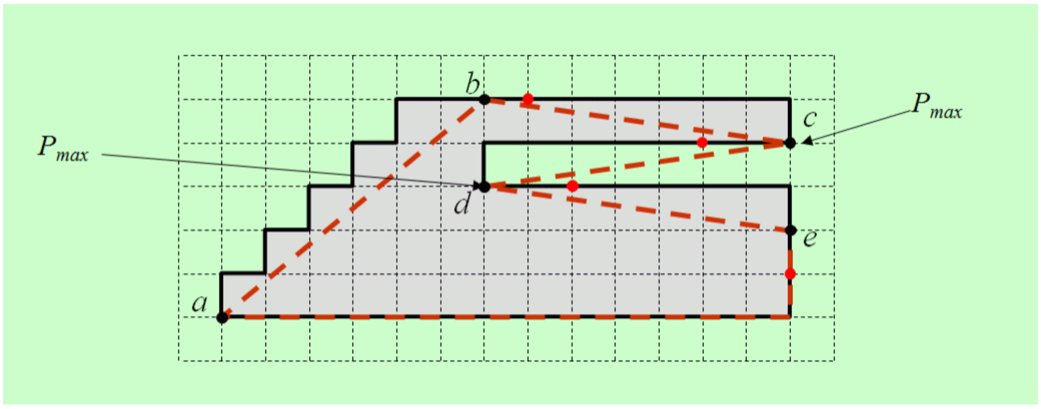
3) If the next point P_i (fig. P_5) lies outside of the new sector, then there is no straight line L through P_1 and P_i such that all points between P_1 and P_i have a distance to L less than ε. Therefore P_i cannot serve as a polygon edge. The point before P_i (fig. P_4) is then the last point of the current segment and the starting point of the next one.



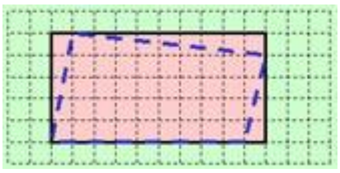
Algorithm of the improved sector method

The sector method is fast because it tests each point only once. However, it guaranties only that the Hausdorff distance between the curve and the polygon is less than the prescribed tolerance ε. The following additional condition for interrupting the current segment of the curve, that should be approximated by the current edge of the polygon, has been introduced. The point P_{max} of the current segment, which has the greatest distance from the starting point of the segment, must be defined during the processing of the segment. If the projection of the actual point P_i onto the line (P_1, P_{max}) is less than $|P_1, P_{max}|-\varepsilon$ then the point P_{max} is the last point of the current segment and the starting point of the next one. This test guaranties that the sequence of the projections of the points of the curve onto the polygon is monotone and therefore the conditions for defining the Schlesingers distance are fulfilled.

In the following example red points are those at which the algorithm comes to the decision to interrupt the current segment. The tolerance ε = 1.5.



Optimal polygonal approximation



An important drawback of the sector method is that it "cuts off" the vertices of right angles with sides parallel to the coordinate axes.
The reason is that it only checks whether the distance of the given points from the polygon is less than the threshold.

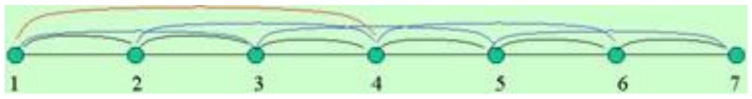
A better method must also minimize the distance of the points from the polygon while retaining the number of polygon edges as small as possible. These contradictive requirements are expressed in the following minimization criterion:

$$F = \sum_{i=1}^{N} (\max_k dist(p_k, E_i) + Penalty \cdot N;$$

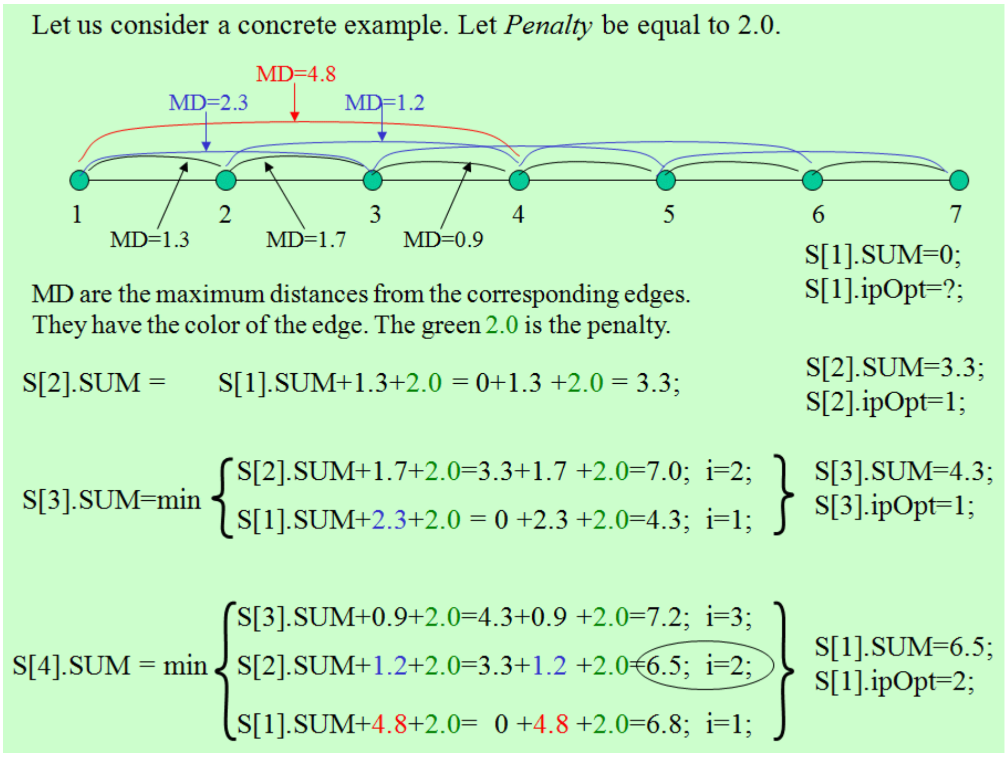
Here is N the number of the polygon edges, "Penalty" is a predefined constant, $dist(p_k, E_i)$ is the distance from the point p_k belonging to the ith segment of the curve to the edge E_i. The maximum is with respect to all points of the ith segment.

The minimization problem can be solved by means of the dynamic programming in the following way. The curve (closed or open) is represented as a sequence of points p_k, $k \in [1,M]$. We allocate an array *State*[M] of structures, where M is the number of points. Each structure *State*[k] contains a float variable *State*[k].*SUM* and an integer variable *State*[k].*iPopt*.

Let us draw the elements of *State*[ ] as a sequence of points (small disks) in the following graph:



The arcs represent possible polygon edges. Consider as an example (the figure below) the point 4 and all possible edges connecting it with some of the points with smaller numbers: the red arc connects the point 4 with the point 1, the blue one - with the point 2 and the black one - with point 3. The arc connecting the point *k* with the point *i*, $i<k$, defines a possible value of the criterion *F*: $F(k)=F(i)+MaxDist(i,k)+\text{Penalty}$, where $MaxDist(i,k)$ is the maximum distance from the points of the segment (*i*, *k*) to the polygon edge *(i,k)*. Thus, if we know the values of *F(i)* for all $i<k$, we can compute the values of *F(k)* corresponding to all arcs having their right ends at p_k. We must find the smallest of these values, save it as *State*[*k*].*SUM* and save the corresponding value of *i* as *State*[*k*].*iPopt*. The value of *State*[1].*SUM* is 0. When starting from *k*=2 one can compute the optimal values for *k*=2, 3,..., *M*. After having reached the last point *k*=*M* one can find its previous optimal point with the index *State*[*M*].*iPopt*, then find its previous point etc. thus reconstructing the optimal sequence of the polygon vertices.



Let us consider a concrete example. Let *Penalty* be equal to 2.0.

S[1].SUM=0;
S[1].ipOpt=?;

MD are the maximum distances from the corresponding edges. They have the color of the edge. The green 2.0 is the penalty.

S[2].SUM = S[1].SUM+1.3+2.0 = 0+1.3 +2.0 = 3.3;

S[2].SUM=3.3;
S[2].ipOpt=1;

$$S[3].SUM=\min\begin{cases} S[2].SUM+1.7+2.0=3.3+1.7+2.0=7.0; & i=2; \\ S[1].SUM+2.3+2.0 = 0+2.3+2.0=4.3; & i=1; \end{cases}$$

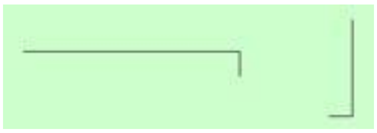
S[3].SUM=4.3;
S[3].ipOpt=1;

$$S[4].SUM = \min\begin{cases} S[3].SUM+0.9+2.0=4.3+0.9+2.0=7.2; & i=3; \\ S[2].SUM+1.2+2.0=3.3+1.2+2.0=6.5; & i=2; \\ S[1].SUM+4.8+2.0= 0+4.8+2.0=6.8; & i=1; \end{cases}$$

S[1].SUM=6.5;
S[1].ipOpt=2;

This method brings the best results as to the precision of the approximation. However, it must be still improved as to its time consuming. One of the ways of doing this consists in subdividing the given curve into longest DSSs and applying the optimization method to the endpoints of the DSSs. However, this approach needs some additional research to find a fast method of estimating the maximum deviation of the points of a DSS from the line connecting its end points.

It is also necessary to eliminate DSSs looking like the following two:



since they will give rise to the above mentioned problem of "cutting off" the vertices of right angles.

The version explained above brings the optimal solution for a fixed starting point of the polygon. The non-trivial, fast solution for the optimal starting point must be still developed. Thus, there is still some research work to be done to bring this method to perfection.