Robust decomposition of a digital curve into convex and concave parts

Tristan Roussillon, Isabelle Sivignon, Laure Tougne
Laboratoire d'InfoRmatique en Image et Systèmes d'information
UMR5205 CNRS/Université Lyon1/Université Lyon2
Université Lyon2 - 5 avenue Pierre Mendes France - 69676 Bron cedex, France
tristan.roussillon@liris.cnrs.fr

Abstract
We propose a linear in time and easy-to-implement algorithm that robustly decomposes a digital curve into convex and concave parts. This algorithm is based on classical tools in discrete and computational geometry: convex hull computation and Pick's formula.

Fig. 1: O (black disks) is bounded by C (squares). (left) The solid line that encloses $CH(O)$ depicts $CH(O)$. (middle) $P \in C$ is 0-convex because $A(L(P)) = 0$. (right) $P \in C$ is neither 0-convex nor 0-concave because $A(L(P)) = 1$ and $A(R(P)) = 2$.

Update of $A(L(P))$ (resp. $A(R(P))$)

Algorithm 1: addPoint(LDeque, leftArea, n, p);
Input: p, the last of the n points of P
Output: $A(L(P))$
1 last = LDeque.back();
2 LDeque.pop_back();
3 prev = LDeque.back();
4 a = A(prev, last, p);
5 while (a < 0) do
6 leftArea += a;
7 last = prev;
8 LDeque.pop_back();
9 prev = LDeque.back();
10 a = A(prev, last, p);
11 LDeque.push_back(p);
12 return leftArea - (n-LDeque.size())/2;

Measures

* convexity($O$) = $\frac{A(CH(O)) - A(O)}{A(CH(O))}$. 
* convexity($P$) = $\frac{A(L(P))/A(CH(O))}{A(CH(O))}$. 
* concavity($P$) = $\frac{A(R(P))/A(CH(O))}{A(CH(O))}$. 

Function A returns the digital area $A(O)$ of a digital object $O$ (i.e., the number of digital points belonging to $O$). The digital area is computed from the Euclidean area thanks to the Pick's formula.

Running of the update algorithm

$A(L(P)) = 2 + 5/2 - 7/2 = 1$
$A(R(P)) = 1.5 + 4/2 - 7/2 = 0$

Binary decomposition

Algorithm 2: AdHocSegmentation(C, k)
Input: A curve $C$ of n points and a threshold k
1 i = 0;
2 while $i < n$ do
3 $P = C$; $i = 0; i++$;
4 while ($A(L(P) \cup C)) \leq k$) and ($i < n$) do
5 $P += C; i +=$;
6 while ($A(L(P) \cup C)) \leq k$) and ($j < n$) do
7 $P += C; j +=$;
8 $F = C_i \cup j; l++;$
9 while ($A(R(P) \cup C)) \leq k$) and ($j < n$) do
10 $P += C; j +=$;
11 while ($A(R(P) \cup C)) \leq k$ and ($j < n$) do
12 $P += C; j +=$;

Results

Conclusion and Perspectives
- Linear in time and easy-to-implement algorithm.
- Can be used to robustly detect digital straight line of any thickness.
- Can be used into a multiresolution framework.