COMPUTATION OF THE HAUSDORFF DISTANCE BETWEEN PLANE VECTOR POLYLINES

J.F. Hangouët
COGIT Laboratory
Institut Géographique National
2, avenue Pasteur
F-94160 Saint-Mandé
France
email : hangouet@cogit.ign.fr

ABSTRACT

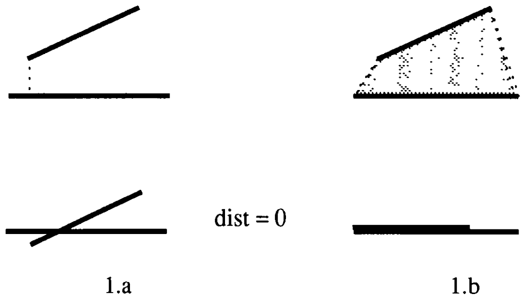
The Hausdorff distance between two objects is a mathematically true distance. When both objects are punctual, it does not differ from the Euclidean distance between points; otherwise it takes into account the mutual positions of the objects relatively to each other. Its main interest for automated cartography, besides quantifying spatial relations between objects, lies in the fact that it expresses *remoteness*. How far are features from each other ? How far is a generalized feature from its original position ? After spotlighting on some properties of the Hausdorff distance applied to geographical features, the paper describes an algorithm for computing the Hausdorff distance in vector mode between two polylines.

KEYWORDS : Hausdorff distance, spatial relations, algorithm.

INTRODUCTION

While the classical Euclidean distance is of foremost importance in surveying issues, GISs have revealed that the point-to-point relation is too limited for cartographic applications. Most geographical features zigzag across the background or swell and bump against each other - they are definitely not points. Distance between features is a difficult concept which has been approximated through various indicators: minimum Euclidean distance [PEUQUET-92], reworked with ε-bands [PULLAR-92], surface "in between" [MCMASTER-92] ... Even if these measures are well adapted to the applications they are meant for (mainly proximity and accuracy evaluations), they all lack at least one of the three prerequisites for being a true mathematical distance : that of separateness, which means that the distance between two objects is zero if and only if those objects are strictly identical (fig. 1). It can be argued that this condition is pointless in cartography, since features will never be strictly the same: attributes or symbolization, if not geometry, will always differ. However, from a geometric point of view, the fulfilment of all criteria for a "gap-quantification" function makes it a safe and systematic distance. The Hausdorff distance is such a mathematical distance - surely not the best, but anyhow a very convenient one.

Figure 1.



Non separateness of some distance functions
1.a minimum Euclidean distance
1.b inner area

HAUSDORFF'S DISTANCE

Definition

Felix Hausdorff (1868-1942) was a German mathematician whose contribution is most remarkable in the field of topology (pure "abstract" topology would be clearer in our GIS and spatial relations context). He built up a distance between objects in finite space as:

$$DH(A,B) = Max\left(sup_{x\in A}\, d(x,B),\, sup_{x\in B}\, d(x,A)\right)$$

where A and B are closed sets and $d\ (x,B)$ the classical Euclidean distance from point x to object B (proof that it is a distance in the mathematical sense can be found in most topology manuals). Two components can be defined (figure 2):

distance from A to B : $D_{A\to B} = \left(sup_{x\in A}\left(inf_{y\in B}\, d(x,y)\right)\right)$

distance from B to A : $D_{B\to A} = \left(sup_{y\in B}\left(inf_{x\in A}\, d(x,y)\right)\right)$

and : $DH = Max\left(D_{A\to B}\, ,\, D_{B\to A}\right)$

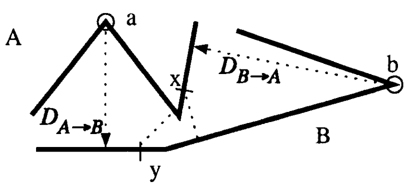


Figure 2.
The two components.
$\forall x \in A,\ d(x,B) \leq d(a,B)$
$\forall y \in B,\ d(y,A) \leq d(b,A)$

The two components are not necessarily of a same value. This is illustrated in the figure above. Other properties of the Hausdorff distance will now be listed, with the examples of polylines, which are closed objects from a mathematical point of view, and a most common representation of geographic features in vector GIS.

Properties

Asymmetry. (fig. 2) The two components usually have different values.

Orthogonality. (fig. 3) The vector representing the Hausdorff component from one object to the other is perpendicular to the second object (or points onto a vertex of the second object). This is a property inherited from the Euclidean distance from point to line.

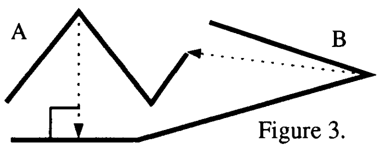
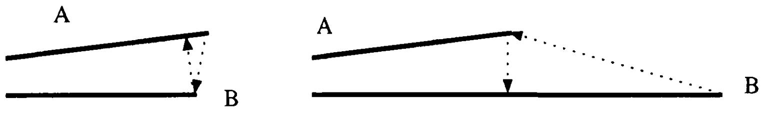


Figure 3.

Sensibility. (fig. 4) Tails make the Hausdorff distance very unstable.

Figure 4.



Tricks. (figure 5) Contrary to intuition, and to what the figures above can suggest, the Hausdorff distance may be achieved between any points of the polylines, and not only on vertices. This makes the computation more difficult.

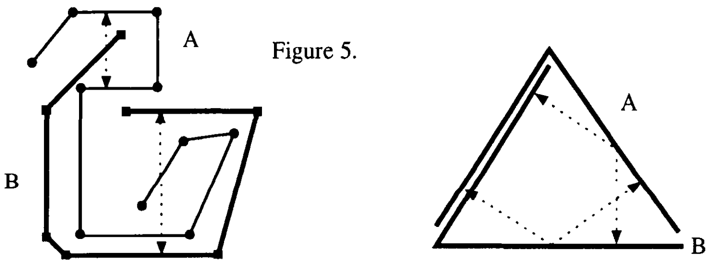


Figure 5.

Tangency. An interesting property of say component *A* to *B* of the Hausdorff distance between polylines is that *when the distance vector starts from a point of A that is not a vertex, there are several distinct points on B which are at the same distance.* In other words, there are several distance vectors, as indicated in figure 5. This result was found while trying to simplify the computation of the distance. An illustration of the proof, rather than the full tiresome demonstration, is given in the appendix.

Applications in automated cartography

The Hausdorff distance between polylines is currently used for feature matching between multi-scale layers [STRICHER-93], for statistical quality controls on linear objects [HOTTIER-92] [ABBAS-94], for the control of generalizing algorithms (current work at our COGIT laboratory).

In cartography, *asymmetry* shows up spectacularly : a generalized line (fig. 6) is closer to the initial line, and the initial line remains far from the generalized line.

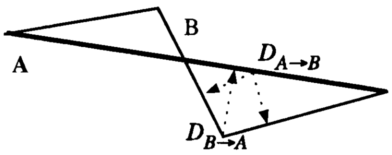


Figure 6. *B is the original line, A its generalization.*
Statistically, [HOTTIER-92],

$D_{A\to B} < D_{B\to A}$

Since all maps and models are generalizations of the real world, Hottier [HOTTIER-92] even states that "maps approach reality but reality remains far from maps" - a truth about resemblance that was already sensed, far from any mathematical justifications, by Edmund Spenser in 1590 [SPENSER-90], in the mouth of the False Fox, to Sir Ape :

And where ye claim yourself for outward shape
Most like a man, man is not like an ape.

Sensibility is a critical issue when comparing objects. The problem is similar to that of the delineation of the area "between" two objects when computing the area distance. Where to cut the lines ? Solutions [STRICHER-93] , [ABBAS-94] are often dependent on the applications.

The tricky aspects of some configurations make the computation of the Hausdorff distance ticklish and time-consuming. Hottier has developed a raster algorithm, and Abbas [ABBAS-94] a vector algorithm, the optimization of which is based on the introduction of a likelihood threshold suited to the statistically expected result. The following algorithm also works on vector polylines, first on the vertices, and then if necessary on the inter-lying segments.

AN ALGORITHM

The two components of the Hausdorff distance may have different values, but the way to compute them is the same. The algorithm given here finds one component, from polyline *A* to polyline *B*. To find the Hausdorff distance, the computation must also be applied reciprocally from *B* to *A*, and the final distance is the greatest of the two results.

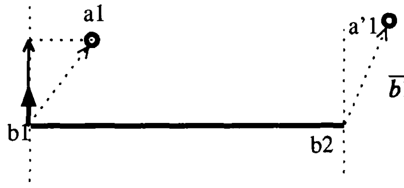
The algorithm to find $D_{A\to B}$ proceeds in three stages :

1. Computation of the distances from the vertices of *A* to polyline *B*.
2. Tests to detect whether further calculation is required or a vertex of *A* bears the greatest distance from *A* to *B*.
3. When no vertex bears the greatest distance, computation of the greatest distance on likely segments.

1. Distances from the vertices of *A* to polyline *B*.

From each segment *sb* = [*b1 b2*] of *B* can be defined a band perpendicular to the segment, with a width equal to the length of the segment (fig. 7). If a vertex *a1* of *A* lies in this band, the distance can be computed from the scalar product between vector **b1 a1** and the unitary vector orthogonal to *sb*. When *a1* lies outside the band, it is closer to a vertex of *B*, with which the Euclidean distance is achieved.

Figure 7



$$\overrightarrow{b1a1}\cdot\overrightarrow{b1b2} \geq 0 \text{ and } \overrightarrow{b2a1}\cdot\overrightarrow{b2b1} \geq 0$$

$$\overrightarrow{b2a'1}\cdot\overrightarrow{b2b1} < 0$$

For a vertex of *A*, the distances to all segments of *B* have to be calculated, the smallest being the Euclidean distance from the vertex to *B*.

2. Check tests.

When computing the distance of each vertex of *A*, a trace must be kept of the component of *B* on which hits the Euclidean distance (a component being either vertex or segment, identified for example by its position number along the polyline : 1st vertex, 1st segment, 2nd vertex, 2nd segment ...). Thus at this stage, all vertices of *A* have their associated components on *B*. The tests will consist in eliminating all segments of *A* which cannot be farther from *B* than their end-points. The tests check all pairs (a1 , a2) of successive vertices of *A*. Such a segment needs no closer analysis if (fig. 8)

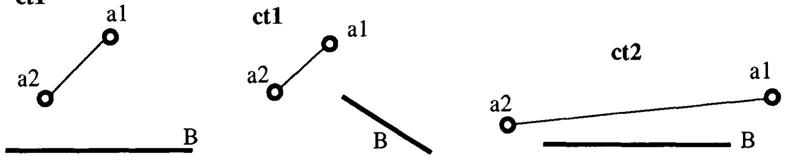
Comp(a1) = Comp (a2) [ct1]

or *Comp(a1) and Comp(a2) are successive vertices of B* [ct2].

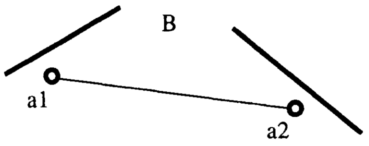
[*Comp(a)* being the number of the component on *B* breeding the Euclidean distance between *a* and *B*].

Such an elimination is justified by the fact that the distance function between two segments is either increasing or decreasing.

Figure 8



This case requires closer observation :



If all successive segments of *A* are thus discarded, it means that the Hausdorff component is achieved from a vertex of *A*, that with the greatest Euclidean distance. Otherwise, further calculation is required.

3. Detail analysis

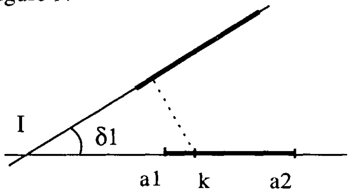
For the remaining segments of *A*, there is suspicion that points between vertices may be farther from *B* than the vertices. For each segment there will be computed the greatest Euclidean distance from *B* - the greatest for all segments being the Hausdorff component.

So now the basic problem is : considering a segment [*a1 a2*] of *A*, the extremities of which are close to different components of *B*, to find the farthest point in-between. For this we will consider all the distance functions to all segments and vertices of *B* from [*a1 a2*], each point *P* on [*a1 a2*] being identified by its k_P parameter so that :

$$\forall P \in [a1a2], \exists! k_p \in [01] \;/\; \overline{a1P} = k_p \overline{a1a2}$$

The distance function d(k) from [*a1 a2*] to a segment is (fig. 9) :

Figure 9.



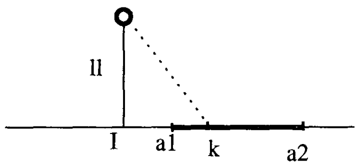
$$d(k) = Abs\left((k - k_I) * d(a1,a2) * \sin\delta_I\right)$$

$$k(a1) = 0$$

$$k(a2) = 1$$

The distance function d(k) from [*a1 a2*] to a vertex is (fig. 10) :

Figure 10.

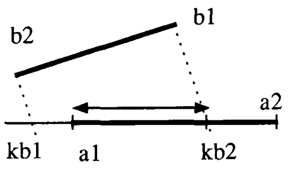


$$d(k) = \left((k - k_I)^2 * d(a1,a2)^2 + ll^2\right)^{1/2}$$

Thus, two numbers have to be computed for each component of *B* : if it is a segment, the k-parameter for the intersection of the two directions, and the sine of the angle; if it is a vertex, the k-parameter of the shortest distance from (a1 a2) to the vertex, and this distance (ll). Let's call this pair of indicators the *distance representation* of the component.

However, it is not necessary to compute all the representations : thanks to the orthogonality property of the Euclidean distance, this computation is required only for the segments which see part of [*a1 a2*] or for the vertex protruding toward [*a1 a2*]. The test at this stage can consist in computing, for each segment of *B*, the k-parameters of the intersections with line (*a1 a2*) of the perpendiculars at both ends : the *effectiveness interval* (fig. 11). The configurations, and the way to recognize them, are described in fig.12.

Figure 11.



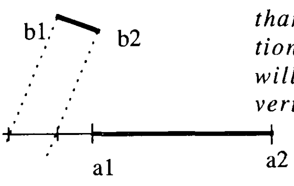
kb is solution of the system :

$$\overline{a1M} = k.\overline{a1a2} \text{ and } \overrightarrow{bM}.\overrightarrow{b1b2} = 0$$

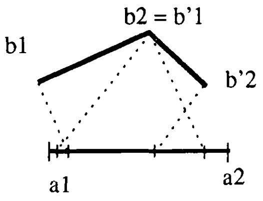
sinδ is the absolute value of the scalar product between a unitary vector of [b1 b2] and a unitary vector orthogonal to [a1 a2].

Let's rename kb1 if necessary, and take kb1 = Min (kb1 , kb2) and kb2 the other one. [Max (0,kb1) Min (1,kb2)] is the effectiveness interval of component [b1 b2].

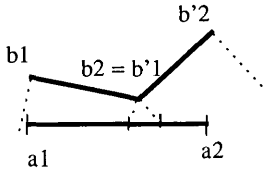
Figure 12.



kb1 and kb2 are both greater than 1 or smaller than 0 : no use trying to find the distance function to segment [b1 b2]. One of the extremities will be nearer anyway, and treated with the vertices.



kb'1 > kb2 : no use trying to find the distance function to vertex b2. Points on [a1 a2] are closer to one of the segments.



kb'1 < kb2 : this vertex points towards [a1 a2], its distance representation has to be kept.

Its effectivness interval is :
[Max (0 , kb2) Min (1 , kb'1)].

So, for one segment [*a1 a2*], the steps above provide a list of potential components on *B*. For each of these, the distance representation has to be computed - and the effectiveness interval stored.

Before describing the algorithm itself, another tool requires description : computation of the intersection between two distance representations.

<u>Intersection of two segment distance representations :</u>

Let the first representation be: (k1, sin δ1), and the second: (k2 , sin δ2).

The resulting *k* are : $k = \frac{k1 - \alpha . k2}{1-\alpha}$ $or\, k = \frac{k1 + \alpha . k2}{1+\alpha}$ where $\alpha = \frac{sin\delta 2}{sin\delta 1}$

If a sine is nul, it has to be checked whether the segment is strictly parallel to [*a1 a2*] or not. If it is, its distance is a constant different from zero, and *k* is easy to find. If the segment is on (*a1 a2*), the common part cannot compete for achieving the greatest distance from [*a1 a2*] to *B*.

<u>Intersection of two vertex distance representations :</u>

Let the first representation be: (k1 , ll1), and the second: (k2 , ll2).

The resulting *k* is : $k = \frac{\frac{ll2^2 - ll1^2}{d(a1\,,a2)^2} + k2^2 - k1^2}{2 * (k2 - k1)}$

If k2 = k1, check if ll2 = ll1. If the two values are different, there is intersection. Otherwise (*a1 a2*) is equally distant from both vertices.

<u>Intersection of vertex - segment distance representations :</u>

(k1, sin d1) is the segment distance representation, (k2 , ll2) is the vertex distance representation. *k* is the root(s) of the following equation :

$$\left[sin^2\delta 1 - 1\right] . k^2 + \left[2 . k - 2 . k1 . sin^2\delta 1\right] . k + k1^2 . sin^2\delta 1 - k2^2 - \frac{ll2^2}{d(a1, a2)^2} = 0$$

<u>Core of the algorithm.</u>

Now that the tools are ready, the algorithm is straightforward :

Start from *a1*. Find its associated component on *B*, its distance function *dfc* and effectiveness interval *eic* [which by construction has 0 for lower bound). We are going in fact to follow the lowest possible path from *a1* to *a2* along *B* (fig. 13).

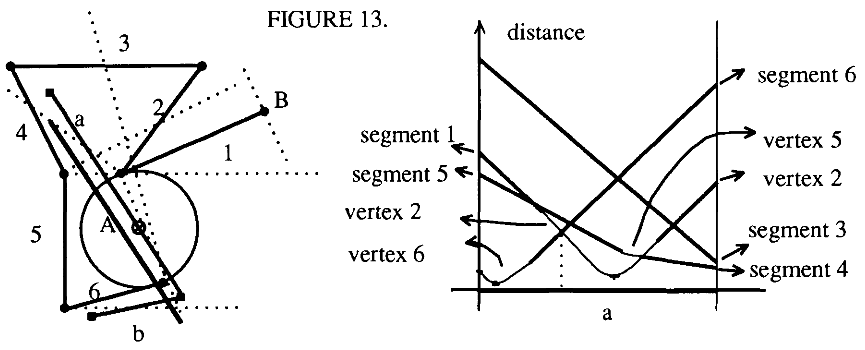
For all other components, find the intersection of their distance function with *dfc*. If the intersection *k* lies without their effectiveness interval or without *eic*, it has to be discarded. For all remaining *k*, the smallest, *kmin*, is the new starting bound. The associated component on *B* is the new component. The search interval becomes [*kmin* , *kb2*] where *kb2* is the upper bound of the effectiveness interval of the new component. The distance between the intersection point and the component has to be stored.

<u>Loop:</u> In this new interval, intersections and new components have to be found on the same criteria.

When no intersection occurs within an interval ending before 1, the following component along polyline *B* has to be found : after a segment, its end-point, after a vertex, the following segment. If the upper bound of the interval is 1, and no intersection occurs, the algorithm has stopped running.

The Hausdorff component is the greatest of all the distance values stored during this pass.

FIGURE 13.



Both extremities of segment b are closer to segment 6 of polyline B : no use investigating on b. However, a has to be looked closely. On the right, the representation of the distance functions from a to the components of B, In this case, there are bound to be intersections, because of the tangency property of the Hausdorff distance. The lowest possible path describes the successive Euclidean distances from a to B; its highest peak is the Hausdorff component.

CONCLUSION

The two components of the Hausdorff distance between polylines give an indication of the mutual remoteness of the polylines, which is a new way of looking at spatial relations. The Hausdorff distance can be an interesting measure on geographical objects - points, lines and contours. The complexity of the algorithm described for finding the component from a polyline with *m* vertices to a polyline with *n* vertices is in most desperate cases $O(m.n^2)$.

REFERENCES

[ABBAS-94] *Base de données vectorielles et erreur cartographique· problèmes posés par le contrôle ponctuel, une méthode alternative fondée sur la distance de Hausdorff: le contrôle linéaire.* IGN, PhD thesis 1994.

[STRICHER-93] *Base de données multi-échelles: association géométrique entre la BD Carto et la BD Topo par mesure de la distance de Hausdorff.* DESS thesis 1993.

[HOTTIER-92] *Contôle du tracé planimétrique d'une carte* Bulletin d'Information de l'Institut Géographique National, Saint-Mandé, France, 1992 pp. 30-36

[PEUQUET-92] *An algorithm for calculating minimum Euclidean distance between two geographic features* Computers & Geosciences Vol. 18 No. 8, 1992 pp. 989-1001

[PULLAR-92] *Spatial overlay with inexact numerical data* Auto-Carto 10, 1992, pp 313-329

[MCMASTER-86] *A statistical analysis of mathematical measures for linear simplification* The American Cartographer Vol. 13 No. 2, 1986 pp. 103-116

[SPENSER-90] *Mother Hubbard's Tale* 1590

APPENDIX : CLUE TO THE TANGENCY PROPERTY

Here is illustrated the fact that when a Hausdorff component does not start from a vertex, it reaches the other polyline in several distinct points.

[$S_{1\,i}$ $S_{1\,i+1}$] is a segment of polyline *A*, and component $D_{A\rightarrow B}$ is achieved between point P1 on] $S_{1\,i}$ $S_{1\,i+1}$ [and point P2 on polyline *B*.

First consequence (A1) : $\forall P \in\ ]S_{1\,i}\ S_{1\,i+1}[\,,\ d(P, B) \leq d(P1, P2)$

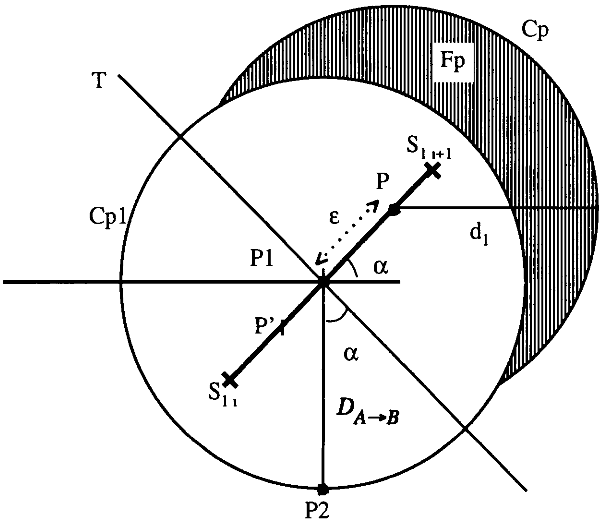
Second consequence (A2) : $\forall Q \in B,\ d(P1, Q) \geq d(P1, P2)$

This means that for every point P on] $S_{1\,i}$ $S_{1\,i+1}$ [, and especially in the most remote part of the segment, there have to be parts of *B* both out of the disc centered on P1, which has $D_{A\rightarrow B}$ for radius (because of A2, disc *Cp1* in fig. 14), and inside the similar disc centered on P (because of A1, disc *Cp* in fig. 14). In other words, there have to be parts of *B* inside the grey crescent *Fp* illustrated in figure 14 below.

This is especially true when P comes close to P1. When P draws on to P1, *Fp* will fuse with the semi-circle of *Cp1* limited by *T*, the perpendicular to ($S_{1\,i}$ $S_{1\,i+1}$) in P1. P2 does not belong to it (P moves on the most distant part of the segment), so there has to be at least one other point of *B* on this semi-circle : one other point at the same distance.

$\alpha = 0$ is a special but not revolutionary case (when dealing with polylines).

Figure 14.


Cp
Fp
T
$S_{1\,i+1}$
P
Cp1
ε
d_1
P1
α
P'
α
$S_{1\,i}$
$D_{A\rightarrow B}$
P2