Computing the Hausdorff Distance Between Curved Objects [1]

Helmut Alt   Ludmila Scharf

Institute for Computer Science, Freie Universität Berlin, Takustr.9, D-14195 Berlin

Key words: shape comparison, Hausdorff distance, parametric curves

1. Introduction

Analysis and comparison of geometric shapes are of importance in various application areas within computer science, such as pattern recognition and computer vision, but also in other disciplines concerned with the form of objects such as cartography, molecular biology, medicine, or biometric signal processing.

The general situation is that we are given two objects A, B modelled as subsets of 2- or 3-dimensional space and we want to know how much they resemble each other [1].

For this purpose we need a similarity measure defined on pairs of shapes indicating the degree of resemblance of these shapes. A frequently used similarity measure is the Hausdorff distance, which is defined for arbitrary non-empty compact sets A and B. It assigns to each point of one set the distance to its closest point in the other and takes the maximum of all these values. Formally, we define the *one-sided Hausdorff distance* from A to B as

$$\tilde{\delta}_H(A,B) = \max_{a\in A}\min_{b\in B} d(a,b), \tag{1}$$

where $d(x,y)$ denotes a distance measure between points x and y.

Here we will assume the planar case, i.e., that $A, B \subset \mathbb{R}^2$ and that d is the Euclidean distance.

The (bidirectional) *Hausdorff distance* between A and B is defined as

$$\delta_H(A,B) = \max(\tilde{\delta}_H(A,B), \tilde{\delta}_H(B,A)) \tag{2}$$

We will only describe the computation of the one-sided Hausdorff distance from A to B, the computation of the bidirectional Hausdorff distance is then straightforward.

[1] This research was supported by the European Union under contract No. IST-2000-26473, Project ECG.

The aim of this paper is to find an algorithm for general shapes which are modelled by two sets of n *algebraic curves*. When we speak about the Hausdorff distance between these two sets we actually mean the Hausdorff distance between the two sets of points lying on these curves. We will restrict to curves that are given by *rational parameterizations*, i.e., each curve is represented by a parameterization

$$c: I \to \mathbb{R}^2, \quad c(t) = (x(t),\ y(t)) \tag{3}$$

where $I \subset \mathbb{R}$ is a closed interval and $x(t), y(t)$ are rational functions with no poles in I.

Observe that this definition includes some important families of free-form parametric curves, for example B-splines.

For simplicity, we assume a certain general position of the input curves. In particular, we assume that any two curves intersect in at most finitely many points.

2. Basic cases

In this section we will investigate how the directed Hausdorff distance between two single objects (curves or points) can be computed.

2.1. *Point-curve and curve-point*

The Hausdorff distance from a point $p = (u,v)$ to a curve $c(t) = (x(t), y(t)), t \in I$ is $\min_{t\in I} d(p, c(t))$.

The Hausdorff distance from a curve $c(t)$ to a point p is the maximum Euclidean distance from any point on c to p.

In order to find those parameters t where the minimum or maximum is attained, we consider the zeroes of the derivative of the squared distance $\frac{\mathrm{d}}{\mathrm{d}\,t}[d^2(p, c(t))]$, i.e., the equation

$$2 \cdot (u - x(t)) \cdot x'(t) + 2 \cdot (v - y(t)) \cdot y'(t) = 0 \quad (4)$$

This equation has constantly many solutions if the degree of c is bounded. We call a point satisfying this equation a *footpoint* of p on c. In addition to the points given by the solutions of equation 4 the minimum or maximum distance can be attained at the endpoints of c (Fig. 1).

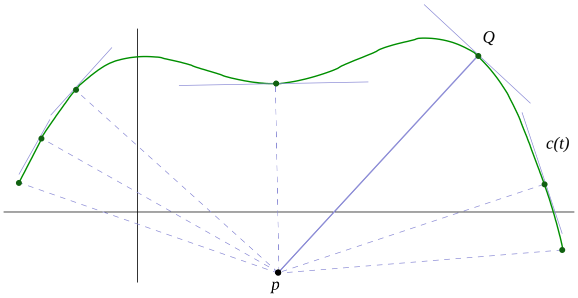

Fig. 1. Hausdorff distance from a curve $c(t)$ to a point p is the distance from p to the farthest point on c.

2.2. *Curve-curve*

We reduce the problem of determining the Hausdorff distance from a curve a, $a(t) = (x_a(t), y_a(t))$, $t \in I_a$ to a curve b, $b(s) = (x_b(s), y_b(s))$, $s \in I_b$ to determining the distances of constantly many candidate points on a to the curve b.

There are four different types of candidate points. Firstly, the Hausdorff distance can be assumed at one of the endpoints of curve a (*type EA*).

Secondly, it can happen that the Hausdorff distance is attained between an endpoint Q of b and a point on a. Therefore, we determine on a all footpoints of the endpoints of b by equation (4) (*type EB*).

For the third type of candidate points we consider the *self-bisector* (or *medial axis*) of a curve, which is the set of all points whose minimal distance to the curve is attained at more than one point on the curve, cf. Fig. 2.

The Hausdorff distance from a to b can be attained at an intersection point of a with the self-bisector of b. We will only give a system of equations describing such a point here for the part of the self-bisector where the two closest points are interior points of b, see Fig. 3.

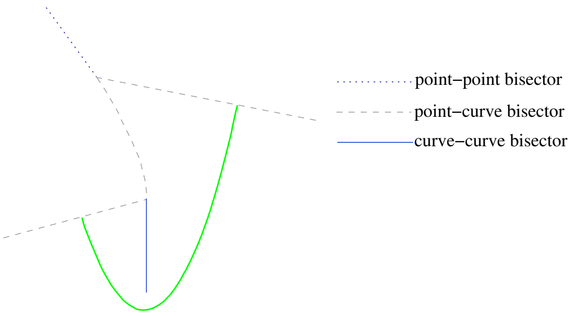

Fig. 2. Self-bisector of a parabola segment.

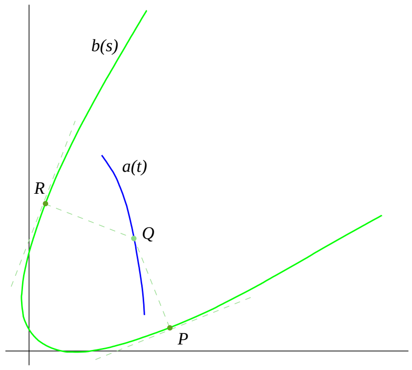

Fig. 3. The Hausdorff distance from the curve $a(t)$ to the curve $b(s)$ is assumed at the intersection point Q of the curve a with the self-bisector of the curve b. The point Q has two different internal foot-points P and R on b.

Suppose $Q = a(t)$ and that $P = b(s)$ and $R = b(r)$ with $r \neq s$ are the points on b closest to Q. Then we obtain the following system of equations for t, s, and r:

$$\left[(x_a(t) - x_b(s))^2 + (y_a(t) - y_b(s))^2\right] - \left[(x_a(t) - x_b(r))^2 + (y_a(t) - y_b(r))^2\right] = 0 \quad (5)$$

$$(x_a(t) - x_b(s)) \cdot x_b'(s) + (y_a(t) - y_b(s)) \cdot y_b'(s) = 0 \quad (6)$$

$$(x_a(t) - x_b(r)) \cdot x_b'(r) + (y_a(t) - y_b(r)) \cdot y_b'(r) = 0 \quad (7)$$

In order to enforce the condition $r \neq s$ we introduce a new variable u and add one more equation to the system:

$$1 - u(s - r) = 0 \quad (8)$$

We add all points determined by the finitely many solutions of this system to the candidate list (*type SB*).

For the fourth type of possible candidate points we observe that the Hausdorff distance can occur between two interior points $p \in a$ and $q \in b$ with-out one of them lying on the self-bisector of the other curve, see Fig. 4.

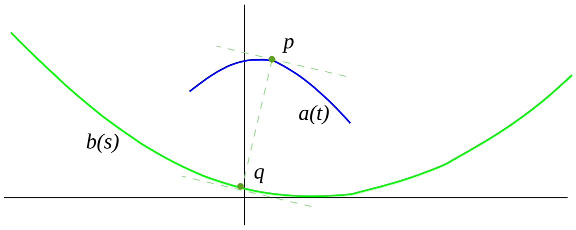

Fig. 4. Hausdorff distance at interior points.

Since q is a footpoint on b for point p, the line segment $\overline{pq}$ must be perpendicular to the tangent line to curve b at point q. In addition it can be shown that the tangent line to the curve a at point p must be parallel to the tangent line to the curve b at point q and, thus, also perpendicular to the line segment $\overline{pq}$. The remaining candidate points (*type I*) for the Hausdorff distance are, therefore, among the solutions of the following system:

$$(x_a(t) - x_b(s)) \cdot x_b'(s) + (y_a(t) - y_b(s)) \cdot y_b'(s) = 0 \tag{9}$$
$$(x_a(t) - x_b(s)) \cdot x_a'(t) + (y_a(t) - y_b(s)) \cdot y_a'(t) = 0 \tag{10}$$

A detailed geometric proof for this fact is omitted due to space limitations and can be found in [7].

3. The general case

As was said before, the general problem we consider is to find the Hausdorff distance between two point sets given by two sets A, B of rationally parameterized algebraic curves.

In order to find $\tilde{\delta}(A, B)$ we split the curves of A at their intersection points with the Voronoi diagram of B. The resulting set A' of curves has the property that each curve lies in one Voronoi cell of B, so its distance to B is the distance to one curve and can be determined using the techniques of section 2.

Computing the complete Voronoi diagram of algebraic curves is a very difficult task in practice and there are some open questions about the bisector of algebraic curves [3–5]. In [5] an approximation algorithm for Voronoi diagrams of curves is given.

Instead, we just compute the intersection points described. The splitting of each curve a of A is done incrementally. Suppose that we have list of intersection points of the Voronoi diagram of a subset of B with a and we want to add a new curve $b \in B$. We do this by scanning the current segments of a. Each segment s belongs to some curve $c \in B$ which has already been processed. First we determine all intersection points of the bisector between b and c with a and find out which ones lie inside s. s is split further with these points and some portions are labelled with b as nearest neighbor, others with c. After this has been done it might be necessary to merge neighboring segments which are both marked with b but are separated by a splitpoint from previous steps.

Similar to the case of self-bisectors, the intersection points of with the bisector between b and c can be found as solutions of systems of equations. We only give this system here for the "interior" bisector of two curves b and c, not the ones for the bisector between an endpoint of one curve and another curve or between two endpoints. These systems, however have to be considered, as well.

The system for the "interior" bisector uses the property that if a point $a(t)$ lies on the bisector between b and c and the corresponding footpoints are $b(s)$ and $c(r)$ then the line segment $\overline{a(t)b(s)}$ is perpendicular to the tangent vector $b'(s)$, and $\overline{a(t)c(r)}$ is perpendicular to $c'(r)$.

$$d(a(t), b(s)) - d(a(t), c(r)) = 0 \tag{11}$$
$$\langle (a(t) - b(s)), b'(s) \rangle = 0 \tag{12}$$
$$\langle (a(t) - c(r)), c'(r) \rangle = 0 \tag{13}$$

The worst case running time of our algorithm as stated is $O(nm^2)$ if A consists of n and B of m curves, assuming that the degrees of the curves are bounded. It can be easily improved to $O(nm \log m)$ by using a divide-and-conquer approach for the splitting of the curves of A. It should be worthwhile to further improve the combinatorial complexity of the algorithm (see also [2]), but our major intent was to find a simple algorithm that can be put into practice with a reasonable amount of effort.

4. Implementation

We implemented the algorithm described in C++ using the computer algebra software library SYNAPS (SYmbolic Numeric APplicationS), see [8,6], for solving the systems of polynomial equations.

For visualization purposes a graphical user interface was developed using the GTK library. It includes visualization of the curves, the input and editing of the parameterization and the intervals if the parameter values, the computation of the Hausdorff distance and the graphical indication of the candidate points considered for the computation. Figure 5 shows the interface with an example.

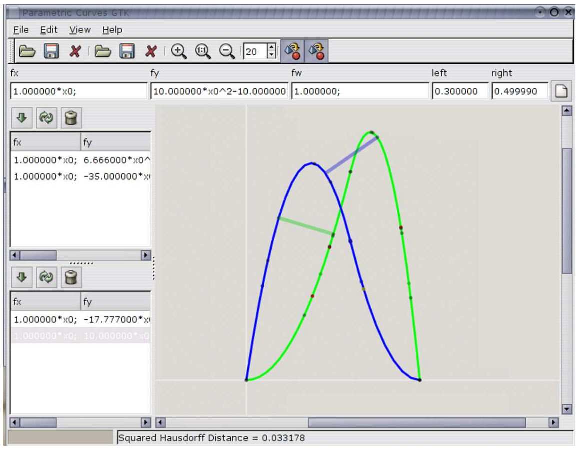

Fig. 5. Result of a Hausdorff distance computation with two B-splines of degree 2. Both one-way distances and all candidate points are shown.

References

[1] Helmut Alt and Leonidas J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of computational geometry.* Elsiever Science B.V., 1999.

[2] Helmut Alt and Otfried Schwarzkopf. The Voronoi diagram of curved objects. In *Proc. 11th ACM Comput. Geom. Symp.*, pages 89–97, Vancouver, BC, 1995.

[3] Gershon Elber and Myung-Soo Kim. Bisector curves of planar rational curves. *Computer-Aided Design*, 30(14):1089–1096, 1998.

[4] Rida T. Farouki and Rajesh Ramamurthy. Degenerate point/curve and curve/curve bisectors arising in medial axis computations for planar domains with curved boundaries. *Internat. J. Comput. Geom. Appl.*, 8:599–617, 1998.

[5] Rida T. Farouki and Rajesh Ramamurthy. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries, I. Theoretical foundations. *Journal of Computational and Applied Mathematics*, 102(1):119–141, February 1999.

[6] G. Dos Reis, B. Mourrain, R. Rouillier, and Ph. Trébuchet. An environment for symbolic and numeric computation. In *Proc. of the International Conference on Mathematical Software*, pages 239–249, 2003.

[7] Ludmila Scharf. Computing the Hausdorff distance between sets of curves. Master's thesis, Freie Universität Berlin, Germany, 2003.

[8] http://www-sop.inria.fr/galaad/logiciels/synaps/.