Hausdorff distance (continued)

Last lecture, we proposed an algorithm for computing the Hausdorff distance between two closed polygonal curves. We will hence call it the I-love-her algorithm for ease of reference. In this lecture, we will focus our attention on some issues of the I-love-her algorithm and other alternatives to compute the Hausdorff distance.

I-love-her algorithm revisited

Pseudo-code for I-love-her algorithm:

1. Main Algorithm

\[
\text{max} = 0 \\
\text{for each vertex } v \text{ in } A \text{ do} \\
\quad \text{find closest point on edge of } B \\
\quad \quad \text{if distance} > \text{max} \\
\quad \quad \quad \text{max} = \text{distance} \\
\text{for each vertex } v_1 \text{ in } B \text{ do} \\
\quad \text{for each vertex } v_2 \text{ in } B \text{ do} \\
\quad \quad \text{for each edge } e \text{ in } A \text{ do} \\
\quad \quad \quad \text{distance} = \text{where } v_1v_2 \text{ bisector intersects } e \\
\quad \quad \quad \text{if min (those distances)} > \text{max} \\
\quad \quad \quad \quad \text{max} = \text{min} \\
\text{for each vertex } v \text{ in } B \text{ do} \\
\quad \text{for each edge } e_1 \text{ in } B \text{ do} \\
\quad \quad \text{for each edge } e_2 \text{ in } A \text{ do} \\
\quad \quad \quad \text{distance} = \text{where parabola intersects } e_2 \\
\quad \quad \quad \text{if min (those distances)} > \text{max} \\
\quad \quad \quad \quad \text{max} = \text{min} \\
\text{for each edge } e_1 \text{ in } B \text{ do} \\
\quad \text{for each edge } e_2 \text{ in } B \text{ do} \\
\quad \quad \text{for each edge } e_3 \text{ in } A \text{ do} \\
\quad \quad \quad \text{distance} = \text{where } B \text{ edges bisector intersects } e_3 \text{ in } A \\
\quad \quad \quad \text{if min (those distances)} > \text{max} \\
\quad \quad \quad \quad \text{max} = \text{min} \\
\text{repeat, swapping } A \text{ and } B \\
\text{return max}
2. How to computer Bisector of vertices u, v?
   construct vector uv
   find the midpoint w of u, v
   rotate uv about w 90 degrees

3. How to computer line-line intersection?
   Jarek’s trick:
   set line 1 in implicit form
   set line 2 in parametric form with parameter t
   put the parametric form in the implicit form and solve for t.

Discussion topics:

1. What is the speed of the algorithm? Is the complexity $O(n^3)$?
2. Can we make I-love-her type of algorithm simpler?

<table>
<thead>
<tr>
<th>Suggestion</th>
<th>Counter Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Will one “me” suffice? Is she always on the middle point if she’s not on a vertex? No.</td>
<td><img src="image1.png" alt="Example" /></td>
</tr>
<tr>
<td>Does the edge-edge case include the other cases? No, as we can see from the diagram, the edge-edge position is different from vertex/vertex position in some configuration.</td>
<td><img src="image2.png" alt="Example" /></td>
</tr>
</tbody>
</table>
Alternative 1: Sampling method

Pseudo-code for Sampling method

Sample the two polygonal curves with sampled vertex on the curve
max = 0
for each sampled vertex a in A do
  a_min_dist = infinity
  for each sampled vertex b in B do
    if(distance(a,b) < a_min_dist)
      a_min_dist = distance(a,b)
    if (a_min_dist > max)
      max = a_min_dist
  repeat with b in B, then a in A
return max

Issues with this method

- The running time now also depends on the sampling rate.
- The error is also related to the sampling rate as shown in the next figure.

In conclusion, there is a tradeoff between the speed (due to sampling rate) and error.
Alternative 2: Voronoi/Delaunay method

Let’s first consider the following puzzle. Suppose Jarek’s wife does not wish to live near a power plant. If we are given all the power plant locations in the United States, is it possible to pinpoint one location that is farthest away from all power plants relative to other locations?

For any triple of power plants, we consider a circle whose perimeter covers all three power plants. If there is not a fourth power plant which situates inside this circle, then we call this triple a Delaunay triangle (Voronoi site): by connecting an edge between two power plants. See picture (thick dots are our power plants).

So why are three vertices necessary? Why can’t we consider only two vertices, or one vertex? To see this, assume that we are to live inside the convex polygon defined by the outermost power plants. Pick any position in the polygon, and consider its distances to all power plants. If this is not the optimal distance, we can certainly keep moving in one direction until we are forced to stop, since now we are equidistant from two vertices. Now our only choice is to move in the direction perpendicular to the line segment connecting the original power plants. We should move in one direction until we again are forced to stop since by continuing, we’re guaranteed to shorten the best distance we could get.

Thus for any triple of power plants, we are able to determine the optimal position within the triangle made up by those plants. If we consider all triples of power plants, and then only consider those which do not have a fourth power plant lying inside each triple, we’d be able to determine the optimal position within each triple. The maximum of all of the distances from these optimal positions to their respective vertices is our desired result.

Suppose now that Jarek’s wife also wants to build the house on the road which is indicated by the red line in the following diagram. Now the possible house site should be the intersection between the road and Voronoi diagram (the blue points).
**Summary of the above paragraphs (Pseudo-code)**

```
best_pos = 0
for all triples of vertices do
    construct circle determine by three vertices
    if there exists a fourth vertex in the circle
        discard this circle
    else
        determine distance from center of circle to vertices
        if distance > best_pos
            best_pos = distance
return best_pos
```

**Voronoi diagrams and another analogy**

Suppose now that the power plants have to shut down due to Jarek’s influence, and instead, post offices are being built on these plants’ old sites. The manager of these post offices would like to know which neighborhood should be served by a particular post office.

If each post office is represented by a bold dot in the figure above, then the dash lines around that particular dot represent the area the post office is responsible for. The area represented by the dash lines is called a Voronoi diagram.

**Hausdorff measure between two polygonal curves**

Note: the material presented in this section can be found in "Approximate Matching of Polygonal Shapes" by Helmut Alt et. al.

Let P and Q be polygons. Define the Voronoi diagram of P to be Vor(P). Vor(P) assignes to each edge and each vertex a region, say e(i) or v(i). This region contains all points which are closer to this edge (or vertex) compared to any other points in the plane.

A lemma in the paper states thus:
**The distance of Q to P (she's on Q and I'm on P) is assumed either at some vertex of Q or at some intersection point of Q with some Voronoi-edge e of P.**

Note: this is a weaker version of the lemma. But this is what was lectured by Jarek, we believed. Hence the Hausdorff deviation between two polygonal curves can be found by taking max{distance of Q to P, distance of P to Q}.