Lecture #8: 22 April 2004
Topics: Shape matching and structural comparison
Scribe: Nina Singhal

1 Introduction

In protein structure comparison, we are trying to determine how similar two structures are. Here, we discuss four methods for measuring the similarity between two geometic shapes.

2 Hausdorff Distance

The Hausdorff Distance is commonly used in computer vision. In that field, a typical problem is that you are given an image and a model of what you want to match to. The goal is to find all the locations in the image which match the model. This is similar to the problem of matching protein motifs within protein sequences. This distance is different from some of the previously discussed measures, because in this case, instead of forming a one-to-one mapping between the two, we allow a many-to-many correspondence. Often times, it is easier to build a many-to-many correspondence, since if we wish to change one assignment, we no longer have a cascade of other assignments which now also need to be redone.

2.1 Mathematical Definition

We are given two point sets $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$ in E^2. The one-sided *Hausdorff distance* from A to B is defined as:

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

The *bidirectional Hausdorff distance* between A and B is then defined as:

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

For fixed A and B, this can easily be computed in time $O((n + m)\log(n + m))$ using Voronoi diagrams. Sometimes, the one-sided distance is preferable, as in the case of partial matching of models to images (under occlusions, etc.).

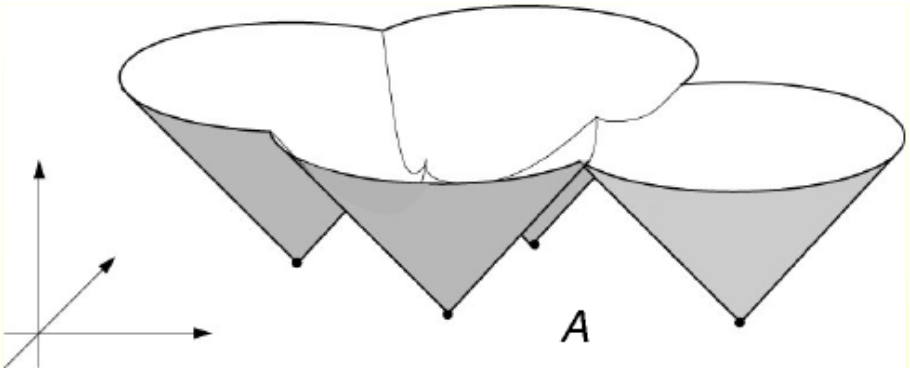

Figure 1: A lower envelope surface

2.2 Variations

In practice, taking the maximum of all the distances is dangerous because possible outliers in one set can then greatly impact the Hausdorff distance. We can compute the *fractional Hausdorff distance*, in which say 90% of the points in A have that distance or less to some point in B.

We can also allow one set of points to be moved by a group of transformations $\mathcal{G}$, for example translations or rotations. This is typically a much harder problem. The distance is then defined as:

$$\tilde{\delta}_{H,\mathcal{G}} = \min_{T \in \mathcal{G}} \max_{a \in A} \min_{b \in B} \|a - T(b)\| \tag{3}$$

2.3 Translation example

Suppose we wish to calculate the Hausdorff distance between A and some translation t of the points in B. Consider the points in A to be points on a plane. The Voronoi surface of A is a conical piecewise surface, where each cone is at a 45° angle to the plane. Since the cones are at 45°, the vertical distance from any point up to the cone is the same as the distance to the apex A. The minimum distance to any point in A can be defined by the lower envelope surface of the cones, shown in figure 1. The distance from any point x would then be

$$d(x) = \min_{a \in A} \|x - a\| . \tag{4}$$

If we consider a translation t of B, then

$$\delta_b(t) = \min_{a \in A} \|a - (b + t)\| = \min_{a \in A} \|(a - b) + t\| = d_{-b}(t). \tag{5}$$

The directed Hausdorff distance between A and some translation t of B is then defined as

$$f(t) = \tilde{\delta}_H(B + t, A) = \max_{b \in B} \delta_b(t) \tag{6}$$

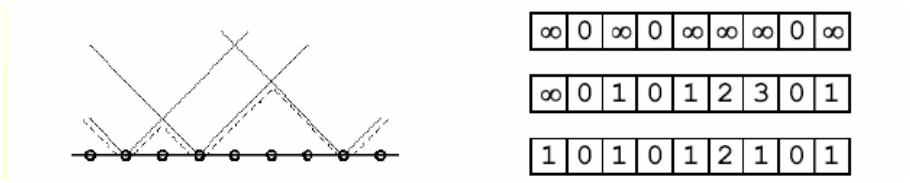

Figure 2: A 1-D example of calculating distance transforms

This is the same as taking the upper envelope of the m Voronoi surfaces, $A - b_1, A - b_2, \ldots, A - b_m$. The running time of an algorithm based on this approach is $O(nm(n + m)\text{polylog}(n + m))$. In general, this doesn't work well unless there are relatively few points. And, if we also allow rotations, or move the points to $3 - D$, the computation time, though still polynomial, becomes too expensive.

However, graphics cards can compute quantized approximations to these lower and upper envelopes using a Z-buffer. These advances in hardware make these methods more practical.

2.4 Raster Hausdorff

It is possible to compute *distance transforms* on a grid given an image. This transform efficiently computes how far each grid point is from the given points in the set. For example, as in figure 2 in 1-D, we can compute this grid in two passes using fast marching or level sets. We begin by initializing all the points in our set to 0 and all other points in the grid to ∞. Then, we do a pass to the right, where we maintain a counter, initially set to ∞. Every step right, we increment the counter, and if we encounter a zero, we set the counter to zero. If the value in the grid is greater than the counter, we set the grid value to the counter value. We follow with a similar pass to the left. In this same manner, we can compute the distance transform for a 2-D set of points by doing 4 passes, $\nearrow$, $\searrow$, $\nwarrow$, $\swarrow$, and in 3-D with 8 passes.

2.5 Fast Hausdorff Search

In practice, it is common to use a branch and bound hierarchical search of the transformation space. If we consider 2-D transformation space of translation in x and y, then the rate at which the Hausdorff distance can change is linear with the translation. So, we can do a *quad-tree* decomposition, where we compute the distance for the transform at the center of each cell. If the distance minus the cell half-width is larger than our current best estimate, then we can rule out that cell; otherwise, we subdivide the cell and consider the children, as in figure 3. A guaranteed, or admissable, search heuristic bounds how good the answer could be in the unexplored region. These search heuristics can't miss any answers, but in the worst case, won't rule out any of the search space. In practice, however, we can rule out the vast majority of transformations. In fact, we can

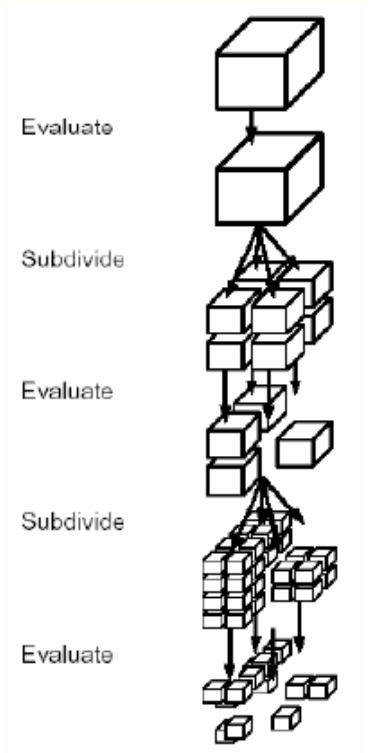

Figure 3: Branch and bound techniques for fast Hausdorff search

use even simpler tests than computing the distance at each cell center.

2.6 Reference Points

For some shapes we can define *reference points* where if you align these points, the rest of the shapes will align reasonably well. These schemes can give constant factor approximations to the Hausdorff distance.

For example, for translation in $2-D$, the lower left corner of the bounding box of the shape serves as a good reference point. Call $\delta = \delta_H(A, B)$ the true Hausdorff distance between two shapes. Also, as in Figure 4, label the lower left hand corner of the bounding boxes of shape A and shape B by r_A and r_B respectively. Then, the maximum x distance between r_A and r_B is bounded by δ, since each point in B is to the right of r_B but each point in A is at most δ away from some point in B, including the point at r_A. The same holds for the maximum y distance between r_A and r_B.

Thus, we can conclude that

$$\|r_A - r_B\| \leq \sqrt{2}\delta. \tag{7}$$

Thus, the Hausdorff distance between A and B', where B' is where we match the lower

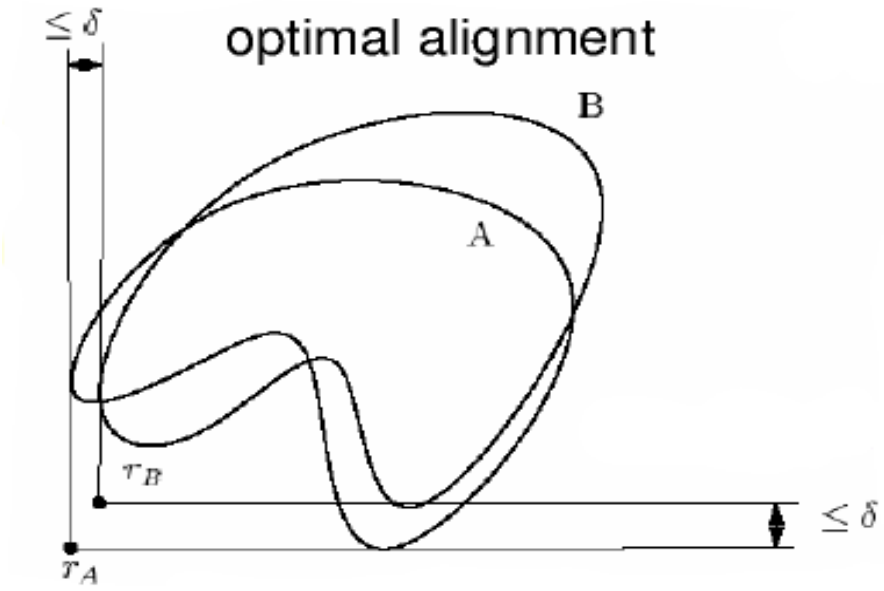

Figure 4: Aligning shapes by their reference points

left bounding box corners is

$$\delta_H(A, B') \leq \delta_H(A, B) + \delta_H(B, B'). \leq (\sqrt{2} + 1)\delta \tag{8}$$

with the first inequality holding because the Hausdorff distance satisfies the triangle inequality. This shows that aligning the bottom left corner of the bounding boxes results in a constant-factor approximation to the true Hausdorff distance. We should also be able to improve on this alignment by local resampling, since we know the optimal alignment is close by.

3 Fréchet distance

One downside of the Hausdorff distance is that it may call things similar which don't seem alike. For example, in figure 5, the two shapes are not alike, but since any point in one is very close to some point in the other, the Hausdorff distance will be small. We may wish for the correspondence between the two shapes to reflect the underlying connectivity of their shapes. The Fréchet distance takes this into account. Imagine a man walking his dog, and the two paths taken by each. We assume that the man and his dog may only move forward on their paths. The goal is to find a mapping through time of the two paths, such that the maximum distance between them is minimum. In other words, we try to find the minimum length leash necessary between the man and the dog. The equation for the Fréchet distance is:

$$\delta_F(f, g) = \inf_{\alpha,\beta} \max_{t\in[0,1]} \|f(\alpha(t)) - g(\beta(t))\| \tag{9}$$

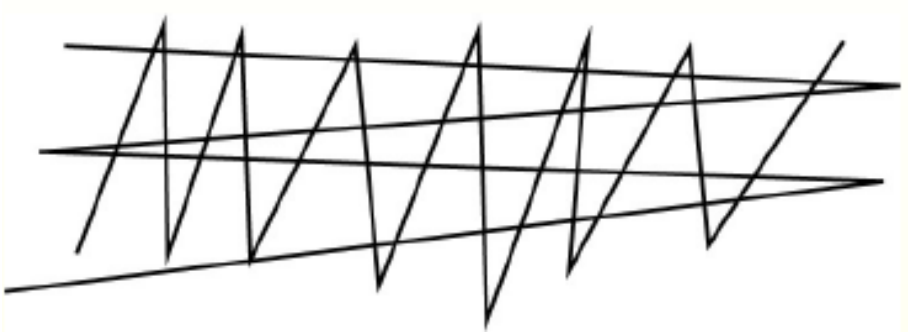

Figure 5: Two different curves

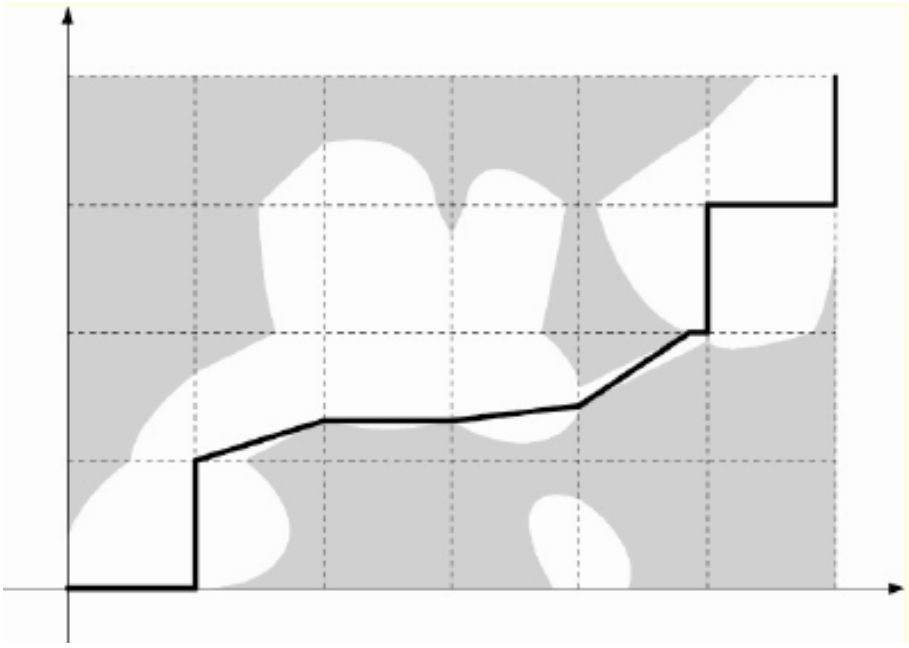

Figure 6: Fréchet decision problem

where f and g are the two shapes and α and β are the two parameterizations.

However, the problem of trying to find a correspondence between the two paths is hard. We can instead switch from the optimatization problem of finding the minimum length of the leash to a decision problem which asks whether a parameterization exists for a leash of a certain length, x. This can be solved by graphing the two paths and coloring in white any points which are within x of each other, as in figure 6. α and β can be any x- and y- monotone paths which connect $(0, 0)$ and (n, m) and always stay in the white regions. Using this decision procedure, we can find the minimum length leash using binary search. The total running time for this algorithm is $O(mn \log(mn))$.

Figure 7: Morphing one shape into another

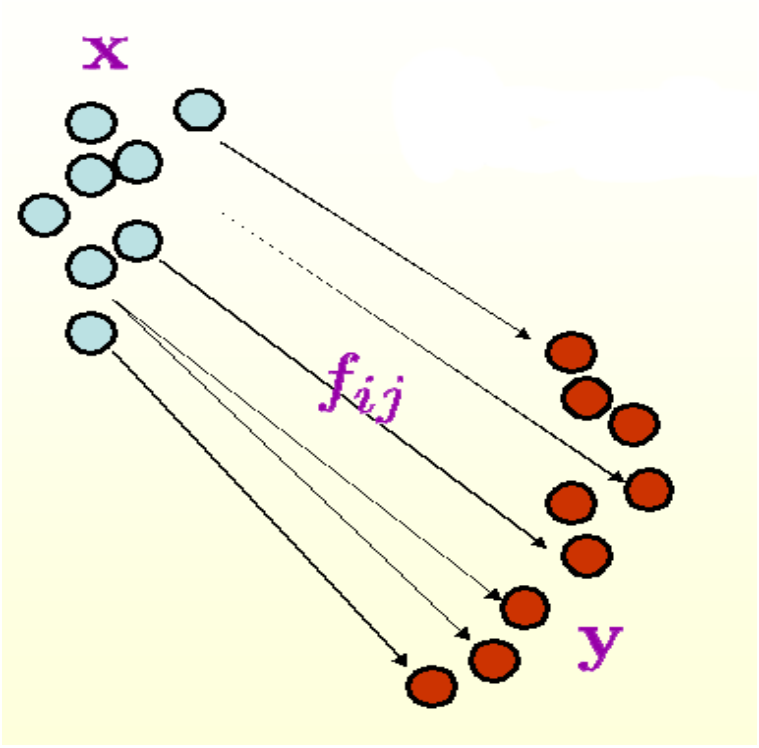

Figure 8: The Earth Mover's Distance

4 Morphing distance

The morphing distance is a measure which computes the cost of changing one shape to another. For example, figure 7 shows how to change a cup to a doughnut through a series of small transformations. This measure also satisfies the triangle inequality.

One example of a morphing distance is the *Earth Mover's Distance*. The problem consists of a set of points x and a set of points y. The goal is to find a matrix f_{ij} which tells how much of the mass of each x_i goes to each y_j, as in figure 8. Mathematically, we are trying to minimize the function

$$\min \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} f_{ij} \tag{10}$$

where c_{ij} is the distance between x_i and y_j, subject to the contraints

$$f_{ij} \geq 0 \qquad i \in \mathcal{I}, j \in \mathcal{J} \tag{11}$$

$$\sum_{i \in \mathcal{I}} f_{ij} = y_j \qquad j \in \mathcal{J} \tag{12}$$

$$\sum_{j \in \mathcal{J}} f_{ij} \leq x_i \qquad i \in \mathcal{I} \tag{13}$$

This can be solved via linear programming. The earth mover's distance is defined as

$$\text{EMD}(x, y) = \frac{\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} f_{ij}}{\sum_{j \in \mathcal{J}} y_j} \tag{14}$$

5 Topological distance

A final way to measure the distance between two shapes is using topological notions. One such notion involves the use of *writhing numbers*, which have previously been used for DNA. The writhing number counts the number of times that pieces of the protein chain cross in front or behind one another. It is defined as

$$W = \frac{1}{4\pi} \int_{S^2} Dw(z) dz \tag{15}$$

where $Dw(z)$ is the count of intersections in the plane normal to the vector z.

References

[1] C. HuttenLocher, G. Klanderman and W. Rucklidge. *Comparing Images Using the Hausdorff Distance*, IEEE Trans. of Pattern Analysis and Machine Intelligence, vol. 15, no. 9, pp. 850-863, 1993.

[2] P.K. Agarwal, H. Edelsbrunner, and Y. Wang. Computing the writhing number of a polygonal knot. *Proceeding Thirteenth Symposium on Discrete Algorithms (SODA)*, 13:791-799, 2002.