Computational Geometry
Point Pattern Matching

Kevin Böckler

17th January 2012

This presentation is based on the work of Alt and Guibas about *Discrete Geometric Shapes* [2].

1 Introduction

Motivation

The Problem. Given two shapes (set of points), $P, Q \subset \mathbb{R}^2$ and a class of allowed transformations T of P, Q, which describes the possible matching techniques. The output in general should be a transformation $f \in T$ which solves one of the following problem: Exact Matching, Approximated Matching or Optimal Matching.

Definition 1. *A transformation f is a function which maps one shape to another: Given $A = \{a \in \mathbb{R}^2 \mid a \in A\}$, then it follows $f : \mathbb{R}^2 \to \mathbb{R}^2$ with transforming the shape: $f(A) = \{f(a) \mid a \in A\}$*

We also define $f(A)$ as transforming a whole set of points A with $f(A) = \{f(a) \mid a \in A\}$
In this presentation we will only look at the following transformations:

Translation : $T = \{f(x) = mx + n \mid m, n \in \mathbb{R}, x \in \mathbb{R}^2\}$

Rotation : $T = \{f(x) = \begin{pmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{pmatrix} x \mid \vartheta \in [0, 2\pi], x \in \mathbb{R}^2\}$

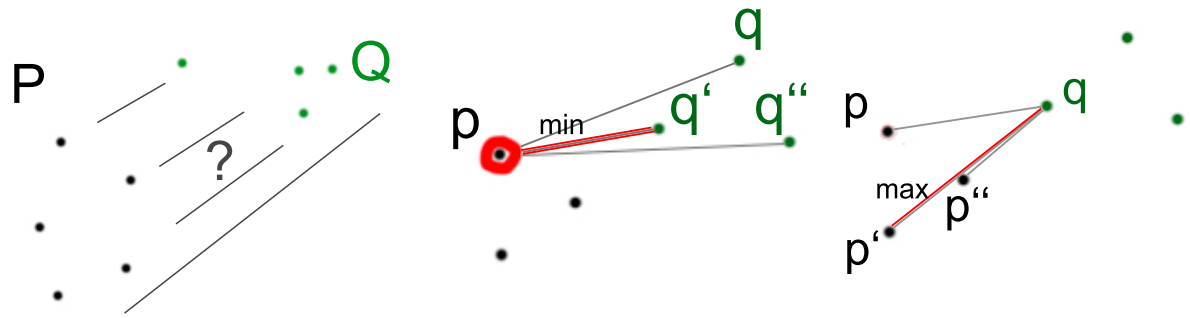
Scaling : $T = \{f(x) = ax \mid a \in \mathbb{R}, x \in \mathbb{R}^2\}$

If we translate and rotate, this is called a *Rigid Motion*, which is a very important class of transformations. Further more, we can also look at rigid motions and scaling, the so called *Similarity*-transformations.

Hausdorff-Distance

When we want to compare two shapes, we look for the distance between them. Our goal is, to choose a distance measurement which, if well chosen, implicates that if the distance of two shapes is small, then both shapes are quite eqal to each other.

Figure 1: Distance of two shapes - Computing the Hausdorff-Distance



Definition 2. *The Hausdorff-Distance is the greatest distance from any point of P to its nearest neighbour of Q:*
$\delta_H(P, Q) = \max_{p \in P} \ \min_{q \in Q} \ ||p - q||_2$

The general problem

We will always somehow describe the matching problems as follows:
Input: $P, Q \subset \mathbb{R}^2$ and T of P, Q
Output: $f \in T$ so that one of the following problems are solved:

> Exact matching: $\delta_H(f(P), Q) = 0$
>
> Approximated matching: $\delta_H(f(P), Q) \leq \varepsilon$, where ε is the allowed error tolerance
>
> Optimal matching: $\delta_H(f(P), Q) = \min_{f' \in T} \delta_H(f'(P), Q)$

In the next chapters we will see, how practicable each of these problems is for real-world applications.

2 Matching of Point Patterns

Definition 3. *Exact matching of two sets P, Q is described by a mapping function $f \in T$, so that $f(p) = q, \forall p \in P, q \in Q$.*
Additionally, P and Q must have the same cardinalities and the mapping function has to be bijective.

A very simple approach for exact matching under rigid motion is given by textpattern-search. To do so, we just parametrize all points of each given shape as a tuple of angle to the centroid and length of this vector. These tuples are sorted and then a matching can be found by checking, whether the sorted sequence of P is somehow contained in Q, so we check $c_P \in c_Q c_Q$.

Algorithm 1 A simple algorithm with textsearch

```
Compute the centroids c_P, c_Q                                   ▷ O(n)
Sort all points of P, Q as pairs of (Φ_i, r_i) and put them into a sequence   ▷ O(n log(n))
if The sorted sequence of P is a cyclic shift of the sequence of Q then   ▷ O(n)
    Just compute the transformation by looking at the first pair of each P, Q   ▷ O(1)
     return this computed transformation
else
    return false
end if
```

Figure 2: Parametrization of point patterns

This algorithm leads to a total runtime of $O(n \log(n))$ due to the sorting complexity. Extending this algorithm with scaling, so we recieve Similarity-Matching, can be simply done by finding a scaling factor from division of both diameters of P, Q. This calculation is done in linear time.

Approximated Matching

Definition 4. *Approximated Matching of two sets P, Q is described by a mapping function $f \in T$, so that $\delta_H(f(p), q) \leq \varepsilon, \forall p \in P$.*

Approximated Matching allows many different perspectives of the problem, so we recieve

One-to-One Matching

Many-to-One Matching

with the question of approximated matching:

Is there a matching $f \in T$ so, that B is matched to A within ε?

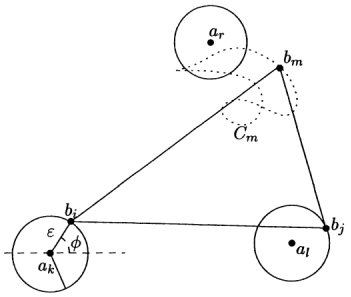
Another formulation of the problem leads to *Optimal Matching*:

For a given P, Q, T: find the smallest ε

One-to-One Matching

Let us now consider a theoretical approach by giving sets of points A, B and look for transformations of the class of rigid motions within an error tolerance of ε. The question is, whether a matching does exist under rigid motions so that all points of B are matched to point of A.

Figure 3: Sets A, B with drawing an algebraic curve by increasing the angle Φ



The idea of this approach is to argue, that if a match of B to A exists, then there will be a matching where two points of B are exactly mapped onto the ε-boundaries of two points of A. Now let these two points travel along the boundaries by increasing Φ (illustrated in Figure 3). At this point, every other point of B is drawing a algebraic curve which somewhere intersects with a ε-boundary of a point of A. With these intersections we can try to find a mutual angle Φ, where all those points lie inside such a boundary. This existence is aquivalent to the problem of finding a perfect matching in a bipartite graph of two sets of points B, A where all points lieing inside a boundary are connected with edges. We then create those bipartite graphs for each angle interval of intersection.

In practice, this approach showed a runtime of $O(n^8)$ just by looking at n^4 different arrangements and then doing some complicated mathematic calculations, which actually are not trivial.

Slightly changing the problem leads to other approaches like

Looking at translation only in $O(n^{1.5} \log(n))$

Assuming there are disjoint ε-neighborhoods of A: $O(n^4 log(n))$

Assuming disjunction and that ε is not too close at the optimal ε: $O(n^2 log(n))$

Many-to-One Matching with respect to the Hausdorff-Distance

If we allow to match multiple points of a set P to one point of Q, then we can use the Hausdorff-Distance to check, whether a matching does well. In fact this means, the smaller the Hausdorff-Distance of tranformed P to the shape Q, the better the transformation is.

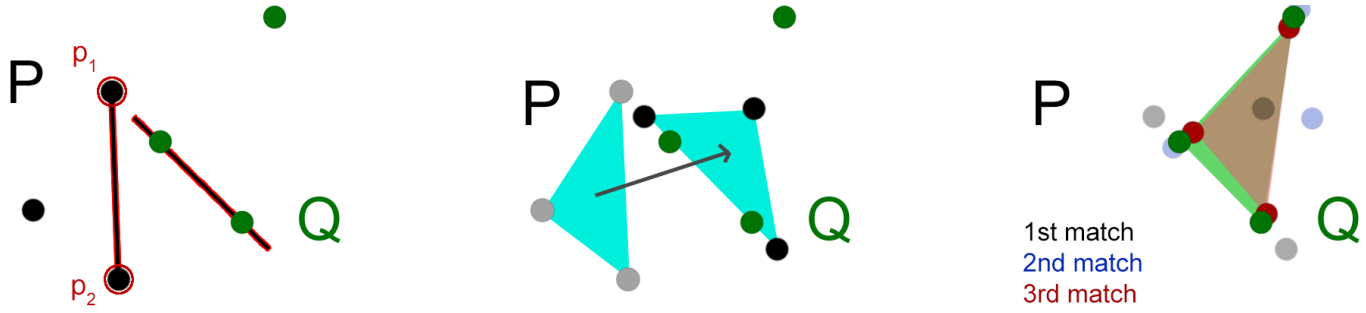
We can calculate the Hausdorff-Distance by just iterating over all points of P and looking at the nearerst neighbour of Q, which leads to a runtime of $O(mn), m = |P|, n = |Q|$. Improving this calculation by using Voronoi-Diagrams:

$O(n \log(n)), n = \max\{|P|, |Q|\}$ for computing the Hausdorff-Distance

Algorithm 2 Matching with respect to the Hausdorff-Distance: Goodrich Approximation

for One diametrically opposing pair of points $p_1, p_2 \in P$ **do** ▷ $O(m)$
  Do a best match to each pair of points $q_1, q_2 \in Q$ ▷ n^2 loops
  Compute the Hausdorff-Distance ▷ $O(m \log(n))$
end for
return The matching with the smallest Hausdorff-distance $T_{\min} = \min_T \ \delta_h(T(P), Q)$ ▷ $O(n^2)$

Figure 4: Goodrich Approximation



This algorithm by Goodrich leads to a total runtime of $O(n^2 m \log(n))$. There is a similar definition of the Goodrich Approximation(see [3]) by not choosing a diametrically opposing pair of points but just taking one point which leads to a smaller runtime (we only do n best matches instead of n^2), but it might lack of effectiveness.

Pattern recognition

Alignment Method

Let S be a scene and M a model, S, M are sets of points. The question now is: *Does the scene S contain the model M?*

Definition 5. *A reference frame of a set M in dependency of two offset points $a, b \in M$ represents a new coordinate system with properties:*

- *a is assigned as the origin* $(0, 0)$
- *b is assigned as an alignment vector* $(1, 0)$
- *all other points relate to these two points*

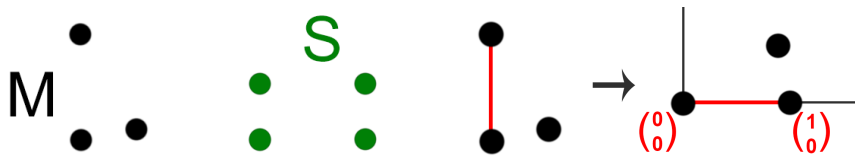
An illustration of a reference frame can be found in Figure 5

Algorithm 3 Alignment Method

for all pairs $a, b \in M$ **do**
  Create a reference frame with offset points a, b
end for
for all pairs $p, q \in S$ **do**
  Create a reference frame with offset points p, q
end for
for all reference frames M_{ref} of M **do** ▷ $|M|^2$ loops
  for all reference frames S_{ref} of S **do** ▷ $|S|^2$ loops
    if All points $m_i \in M_{\text{ref}}$ lie in an ε-neighborhood of a point of S_{ref} **then** ▷ $O(|M| \log(|S|))$
      return true
    end if
  end for
end for
return false

The last nested loop dominates the runtime with: $O(|M|^3 |S|^2 \log(|S|))$.

Figure 5: A Scene S and a Model M. Creation of one reference frame of M



Hashing

In the case we give more than one model or scene and therefore want to compute, which one is at most the given scene or model, we use *Hashing*. A simple example of application would be OCR-technique, where one letter should be found out of a list of available letters. Extending the Alignment Method so that locations of points are saved as hashkeys in a hashtable of reference frames allows so called *Voting*.

First all reference frames for the list of letters are created and put into a hashtable. Then, compute the reference frame for the single letter you want to find out and just compute the hashkeys of its points. For each hashkey simply increment a counter in the hashtable and take the entry with the most votes.

3 Matching of Curves & Areas

In the following section we try to find appropriate techniques to match lines, curves and areas which, obviously, have an infinite amount of points.

Figure 6: Matching of line segments

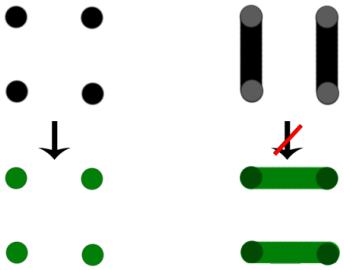
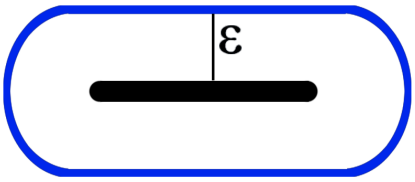


Figure 7: A line segment with a racetrack of ε around it



We will start by looking at a theoretical approach for matching shapes under translations consisting of line segments: Let $A = \{(a, b) \mid a \in \mathbb{R}^2 \wedge b \in \mathbb{R}^2\}$ be a set of line segments.

Definition 6. *A Racetrack A_ε is a disk of radius ε around a given line segment $A_\varepsilon = A \oplus C_\varepsilon$, where C_ε is a circle with radius ε (see Figure 7)*

Now let A_i^ε be the intersection of A_ε and the *translated racetrack* A_ε by a vector b_i
$A_i^\varepsilon = A_\varepsilon \cap (A_\varepsilon \oplus b_i)$ (see Figure 8).

Theorem 1. *There is a matching exactly if* $S = \bigcap_{i=1}^{m} A_i^\varepsilon \neq \emptyset$.

This theorem is aquivalent to the following statement: *It exists a cell of depth m.*

That means a match can be found if and only if all translated racetracks of A along all line segments $b_i \in B$ intersect at least at one point. This suggests, that the given shapes A, B are quite close to each other. If we look at a simple example (Figure 9), one can see, that A and B can be matched under the given ε. If we would now rotate B orthogonal to A, then there would no longer be an area of intersection $\Rightarrow$ A, B could not be matched any longer.

Computing the cells and therefor look for a cell of depth m can be done by applying a line sweep algorithm. Since there are $(mn)^2$ intersection points of the arrangement, the whole algorithm is dominated by the sweep algorithm of runtime:
$O((mn)^2 \log(mn))$

Figure 8: Intersection A_i^ε of two racetracks

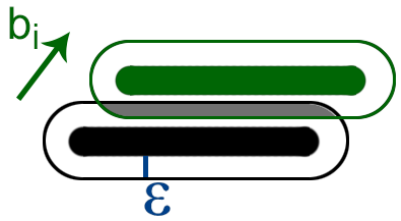
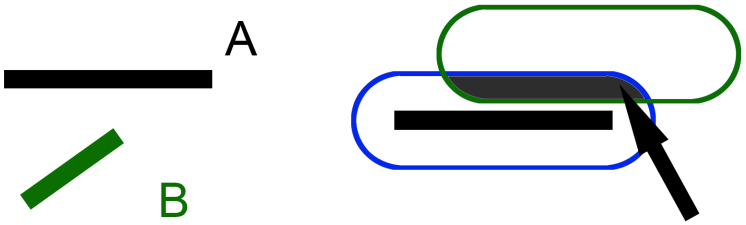


Figure 9: Matching of two simple line segments



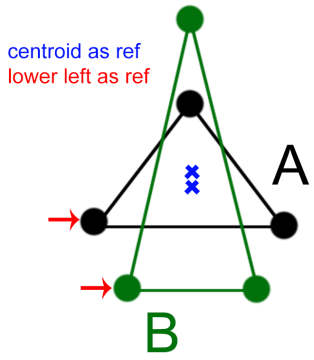
More practicable approach

Since the approach described before is really complicated and rigid motions would as well increase the running time and complexity of implementation, we will now view a technically more promising idea.

Definition 7. *A reference point r_A is a representive point of a shape A so that a perfect matching of A to B bounds the distance of $T(r_A)$ to r_B by a constant factor a, which is called the* quality of the reference point

The quality exactly fulfills the following equation: $a\ \delta(T_{\text{perfect}}(A), B) = \delta(T_{\text{perfect}}(r_A), r_B)$
$\Leftrightarrow a = \frac{\delta(T_{\text{perfect}}(r_A), r_B)}{\delta(T_{\text{perfect}}(A), B)}$

Figure 10: Matching under reference points



The idea of matching A to B is now, that we somehow compute the reference points r_A, r_B and simply find a matching transformation of those points. This transformation is then used for transforming A to B.

First finding a reference point is quite easy for translations only. One can show, that taking the lower left point of a bounding box of the convex hull of A, B provides a quality $a = \sqrt{2\pi}$. Under rigid motions the task of finding a very good reference point is more complex.

Theorem 2. *Translating A to B by matching r_A to r_B is at most $a+1$ times worse than the optimal matching*

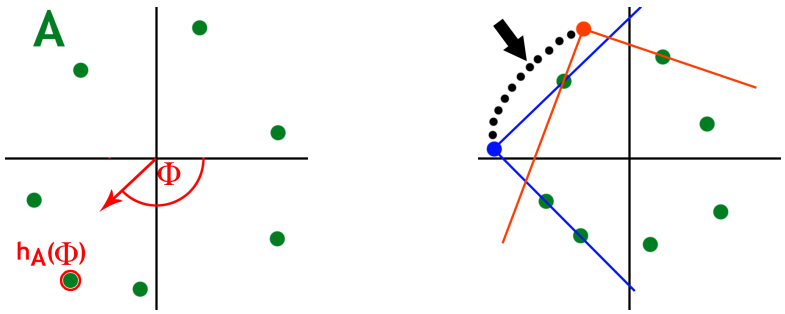
Since finding a reference point for translations as above needs linear time, we recieve a very good translation algorithm with runtime of $O(n)$, because computing the translation vector for both reference points is done in constant time.

Rigid motions: Steiner point

If we want to match two shapes under rigid motions with respect to their reference points, we first translate r_A to r_B and then rotate. Because using obvious reference points based on the convex hull of a shape leads to $a \approx 16.6$ under rigid motions, we now consider a better (and actually the best) reference point.

One can show, that the steiner point is the best reference point for translations. But also for ridig motions the steiner point reaches a quality of $a \approx 1.27$ in $2D$ and $a \approx 1.5$ in $3D$. In addition, the steiner point also deals with similarity transformations. So allowing the class of scaling with rigid motions does not decrease the quality of this reference point.

Figure 11: Steiner Point geometric idea



The input for computing the steiner point always has to be a convex body. The idea of the steiner point for now is, that we extend the approach of the lower left corner of the convex boundary. Now we rotate this rectangular bounding box of the convex body around the given shape along the angle from 0 to 2π. The reference point then is an average value calculated of all these bounding boxes (the lower left corner). We can see this rotation in Figure 11(right).

The left image of Figure 11 shows the help function h_A: it just describes the point, which lies most in the direction of the given angle Φ. Simplifying this approach provides us with the computation of the steiner point $s(A)$:

Definition 8. *The Steiner Point is a reference point of quality* $a = \sqrt{2/\pi}\sqrt{d+1}$*, where* d *is the dimension.*

$$s(A) = \frac{1}{\pi} \int_0^{2\pi} h_A(\Phi) \begin{pmatrix} \cos\Phi \\ \sin\Phi \end{pmatrix} d\Phi$$

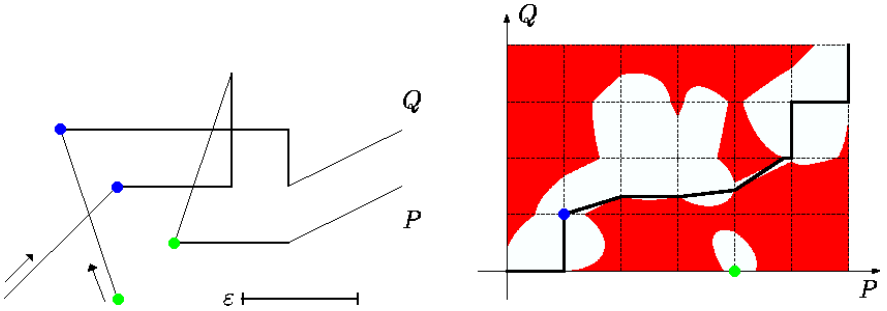
More analysis about the steiner point compared to other reference points are given by Alt, Aichholzer and Rote[1].

Fréchet Distance

While the Hausdorff-Distance was very useful if looking at sets of points, it fails when analysing more complex shapes given by line segments or even more arbitrary curves and areas. A distance between two points now does not always have to be the shortest euclidean path. If we look at a map of streets for example, then the shortest distance is measured by finding the shortest path in a graph of line segmenst. Computing the Hausdorff-Distance would just not work.

Definition 9. *The Fréchet Distance is the greatest distance which can appear when walking along two monoton paths through both given shapes.*

Figure 12: Calculating the Fréchet Distance using a P/Q - diagram



The idea for computing the Fréchet Distance between polygonal curves P, Q (see also Pelletier[4]) is to parametrize both curves and represent all possible configurations of monotone walking in a so called P/Q - diagram. For all configurations where the euclidean distance between two current points of P, Q is at most ε - the given distance -, we draw an area of allowed space (white space in the figure).

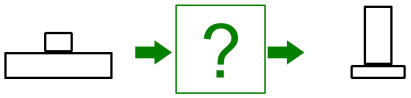
Under the given input P, Q and the distance ε we can now check, if the given polygonal curves have a Fréchet Distance of at most ε by solving the decision problem, whether to find a monotone path through the P/Q - diagram (moving through allowed space only).

One can clearly see, that the complexity of this algorithm is quadratic in steps of parametrization, or more precise: $O(mn)$, where $m = |P|, n = |Q|$.

4 Outlook

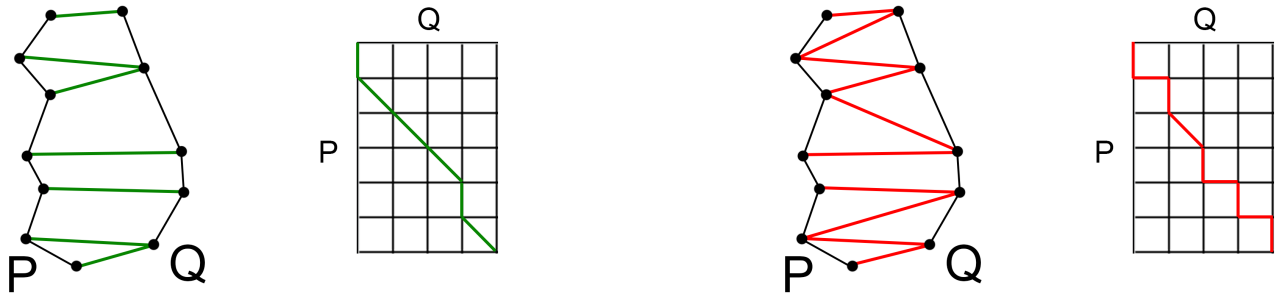
There are some more topics how to work with shapes such as interpolation two shapes into each other or - very useful - simplificating shapes so the input size can be reduced. You can read more of these topics in the work of Alt and Guibas[2].

Figure 13: Interpolating two shapes



Interpolation of polygonal shapes can be achieved by simply defining one point as starting point and then iterate over all points - in a monotone direction - trying to somehow match these points to those of the other shape.

Figure 14: Interpolating polygonal chains



Simplification of shapes

When trying to simplificate a given shape, we can seperate this process into two types: First, given a set of points A, then we try to find a subset of points $A' \subset A$. The property of new set should be, that it has a significant smaller cardinality while keeping the resulting shape as equal to A as possible.

Second, a shape can be simplificated by finding arbitrary points which represent the original shape. As expected, this method is much more difficult to handle in practice.

References

[1] Alt, H., Aichholzer, O., and Rote, G. Matching shapes with a reference point. In *Symposium on Computational Geometry* (1994), pp. 85–92.

[2] Alt, H., and Guibas, L. J. *Discrete Geometric Shapes : Matching , Interpolation , and Approximation.* 1999, ch. 3, pp. 123–151. HANDBOOK OF COMPUTATIONAL GEOMETRY.

[3] Goodrich, M., Mitchell, J., and Orletsky, M. Approximate geometric pattern matching under rigid motions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 4 (Apr. 1999), 371–379.

[4] Pelletier, S. CS507 Project - Computing the Fréchet distance between two polygonal curves. `http://cgm.cs.mcgill.ca/~athens/cs507/Projects/2002/StephanePelletier/`. [Accessed: 09/11/2012].