INSTITUTE OF
COMPUTER SCIENCE
FREIE UNIVERSITÄT BERLIN

Bachelor Thesis

Lexicographic Fréchet Matchings with Degenerate Inputs

Anton Irmfried Norbert Begehr
Matriculation-Nr: 5013449
a.begehr@fu-berlin.de

Supervisor: Prof. Dr. Günter Rote

Berlin, July 6, 2018

Abstract

Ok.

Statutory Declaration

I declare that I have developed and written the enclosed Bachelors Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The Bachelors Thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

Anton Begehr
Berlin, July 6, 2018

Contents

1 Introduction

1.1 Motivation

!!!TODO!!! compare two paths

1.2 Applications

!!!TODO!!! This is a short overview of real world applications of the Fréchet Distance.

1.2.1 Molecule Chains (Proteins)

1.2.2 Handwriting Recognition

1.2.3 Comparing Routes

1.2.4 Computer Vision

1.2.5 Wifi access point placing

1.3 Problem Metaphor (Dog Walking)

The Fréchet distance is often visually explained using the "dog walking"-metaphor, which is described in the following paragraphs.

Imagine two curves:

- Curve C_P tracing the walking curve of a man
- Curve C_Q tracing the walking curve of the man's dog

The Fréchet Distance then is the shortest possible length of a leash, the man would need to keep ahold of his dog. Both the man and his dog have to traverse their entire paths and neither are allowed to walk backwards.

Both the man and his dog can independently vary their respective speeds. With the conditions described above, at every point in time, either both the man and his dog are walking forwards along their respective paths or one is idle while the other is walking along his path.

To make calculating the Fréchet distance more feasible by simplification, the arbitrary curves C_P, on which the man is walking, and and C_Q, on which his dog is walking, are approximated each by a connected chain of line segments, a polygonal chain. The arbitrary curve C_P is approximated by the polygonal chain P. The arbitrary curve C_Q is approximated by the polygonal chain Q.

2 Preliminaries

2.1 Polygonal Chains

Given are two polygonal chains P and Q. A polygonal chain is defined by a list of points, sorted by the order of the chain, ascending from beginning to end. The first point of the list is the start of the polygonal curve and the last point of the list is the end of the polygonal curve. This order implies the direction of the directional polygonal curve.

P and Q are polygonal chains consisting of n and m points respectively.

$$P_{points} = [P_1, P_2, \dots, P_n]$$
$$Q_{points} = [Q_1, Q_2, \dots, Q_m]$$

Points of P and Q are expressed as vectors:

$$P_i = \begin{pmatrix} x_{Pi} \\ y_{Pi} \end{pmatrix}, Q_j = \begin{pmatrix} x_{Qj} \\ y_{Qj} \end{pmatrix}$$

For simplification we assume all points P_i of P and Q_j of Q lie in the xy-plane, but the results generalize easily to arbitrary dimensions[2].

2.2 Parametrisation

We will use an arc-length parametrisation to define the polygonal curves P and Q going forward. We are using the arc-length parametrisation, instead of an unit-length parameter, to achieve visualisations with proportional scaled cells in the further course of this examination.

First we calculate the euclidian distances between the points of P and Q in the order they appear:

$$d_{Pi} = \|P_{i+1} - P_i\|, 1 \leq i \leq n-1$$
$$d_{Qj} = \|Q_{j+1} - Q_j\|, 1 \leq j \leq m-1$$

We also calculate the distances L_{Pi} and L_{Qi}, that define the distance from beginning of P and Q to the point P_i and Q_j on the P and Q respectively:

$$L_{Pi} = \sum_{k=1}^{i} d_{Pk}, 1 \leq i \leq n-1$$

$$L_{Qi} = \sum_{k=1}^{j} d_{Qk}, 1 \leq j \leq m-1$$

The total lengths of P and Q are then defined as $L_P = L_{Pn-1}$ and $L_Q = L_{Qm-1}$.

We apply an arc-length parametrisation on P and Q using parameters $x \in [0, L_P]$ and $y \in [0, L_Q]$ respectively. These parametrisation can be expressed as follows:

$$P(x) = \begin{cases} P_1 + \frac{x}{d_{P1}}(P_2 - P_1) & 0 \le x \le L_{P1} \\ P_2 + \frac{x - L_{P1}}{d_{P2}}(P_3 - P_2) & L_{P1} < x \le L_{P2} \\ \dots & \\ P_{n-2} + \frac{x - L_{Pn-2}}{d_{Pn-1}}(P_{n-1} - P_{n-2}) & L_{Pn-2} < x \le L_{Pn-1} \end{cases}$$

$$Q(y) = \begin{cases} Q_1 + \frac{y}{d_{Q1}}(Q_2 - Q_1) & 0 \le y \le L_{Q1} \\ Q_2 + \frac{y - L_{Q1}}{d_{Q2}}(Q_3 - Q_2) & L_{Q1} < x \le L_{Q2} \\ \dots & \\ Q_{m-2} + \frac{x - L_{Qm-2}}{d_{Qm-1}}(Q_{m-1} - Q_{m-2}) & L_{Qm-2} < x \le L_{Qm-1} \end{cases}$$

2.3 Height Function

To calculate all possible distances between points on P and Q, we will define the height function δ that maps each point pair R consisting of one point on P and one point on Q to the euclidian distance between the points.

$$R = [0, L_P] \times [0, L_Q]$$
$$(x, y) \in R$$
$$\delta(x, y) := \|P(x) - Q(y)\|$$

3 Classical Fréchet Distance

This section introduces the Fréchet distance and gives an overview of the main findings in the 1995 paper "Computing the Fréchet distance between two polygonal curves" by Alt and Godau[1], which later will be essential for computing the lexicographic Fréchet distance.

3.1 Traversal of P and Q

Let us revisit the "dog walking"-metaphor (Section 1.3) of the man and his dog walking along, or in other words traversing, their paths. The Fréchet distance is a dissimilarity measure that considers the traversal of both polygonal chains P and Q with following conditions:

(A) Both P and Q are traversed completely.

(B) Both P and Q are traversed monotonically ascending in the direction of P and Q respectively. Meaning a traverser on either P or Q can stand idle and go forwards, but may never go backwards.

We traverse both P and Q together. This means that at every point in time one of the following cases holds true:

- Both the traverser on P and the traverser on Q are moving.
- The traverser on P is moving and the traverser on Q is standing idle.
- The traverser on Q is moving and the traverser on P is standing idle.
- Neither the traverser on P, nor the traverser on Q is moving.

The last case is identical to the end of both traversals. Both traverses have reached the end of their polygonal chains.

We define a traversal of both P and Q together as a joint parametrisation $(\alpha(t), \beta(t))$. This joint parametrisation represents a continuous curve in the parameter area R. The curve represented by $(\alpha(t), \beta(t))$ is a monotonic curve, because of above condition (B) which states that both P and Q are traversed only forwards and never backwards.

Revisiting the "dog walking"-metaphor from Section 1.3, the Fréchet distance is the length of the shortest leash, with which it is possible for the man and his dog to traverse their paths completely by only travelling forward. In other words, the Fréchet distance is defined as the largest distance between the traverser on path P and the traverser on path Q of the traversal with the smallest maximum distance between the respective traversers.

Using the joint parametrisation $(\alpha(t), \beta(t))$ as representative for a traversal, next the distance function $f(t)$ for the joint parametrisation needs to be defined to determine the Fréchet distance. The height function defined in Section 2.3 will be used to define the distance function $f(t)$ for a joint parametrisation $(\alpha(t), \beta(t))$:

$$f(t) = \delta(\alpha(t), \beta(t)) := \|P(\alpha(t)) - Q(\beta(t))\|$$

Now, the Fréchet distance of P and Q is the largest value of $f(t)$ for the joint parameterisations $(\alpha(t), \beta(t))$ with the smallest maximum of all possible joint parameterisations.

3.2 Free-Space Diagram

The free-space diagram F_ϵ is the set of all points of R that lie below the given height ϵ:

$$(x, y) \in R$$

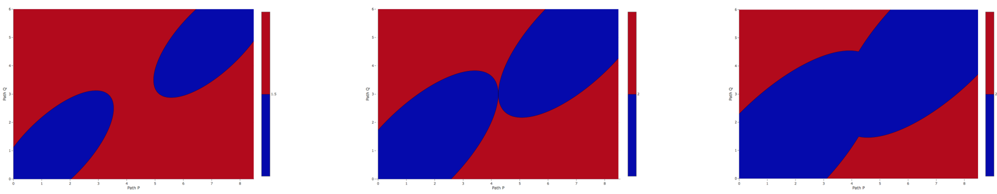
$$F_\epsilon = \{(x, y) \mid \|P(x) - Q(y)\| \leq \epsilon\}$$

The fundamental insight of Alt and Godau[1] is that the Fréchet distance of P and Q is the smallest ϵ for which a monotonic path exists from $(0, 0)$ to (L_P, L_Q) in F_ϵ.[2]

3.3 Decision Problem

The decision problem asks if a height function δ can be traversed monotonically while not exceeding the height of a given ϵ. In their paper, Alt and Godau[1] define the decision algorithm, which solves the decision problem by using the free-space diagram F_ϵ.

In the following example we will see three free-space diagrams F_ϵ for the same height function with different ϵ. ϵ_F denotes the Fréchet distance.



(a) $\epsilon < \epsilon_F$ (b) $\epsilon = \epsilon_F$ (c) $\epsilon > \epsilon_F$

Figure 1: Three free-space diagrams with same height function δ[1]

The following cases are depicted in Figure 1 above:

(a) ϵ is smaller than the Fréchet distance ϵ_F. The decision algorithm returns false.

[1] View the example here: `https://abegehr.github.io/frechet/?p=(1_2)(4_5)(7_2)&q=(1_3)(7_3)`

(b) ϵ is equal to the Fréchet distance ϵ_F. The decision algorithm returns true.

(c) ϵ is larger than the Fréchet distance ϵ_F. The decision algorithm returns true.

3.4 Classical Critical Events

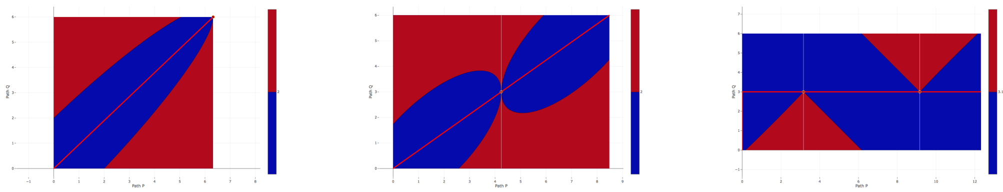
Alt and Godau also identified, that the critical height ϵ_F, where a critical pathway in the free-space diagram closes, occurs in one of three cases. These are the three types of classical critical events:

(a) ϵ_F at $(0, 0) \in F_\epsilon$ or $(L_P, L_Q) \in F_\epsilon$

(b) A new passage with height ϵ_F opens at border between two cells.

(c) A new horizontal or vertical passage with height ϵ_F opens on borders with cells in between.

In Figure 2 example free-space diagrams for each type of classical critical event are shown.



(a) type a[2] (b) type b[3] (c) type c[4]

Figure 2: Examples for the three types of classical critical events

[2]Classical critical event of type a at height ϵ_F is at the upper-right corner (L_P, L_Q). View the example here: `https://abegehr.github.io/frechet/?p=(1_3)(7_5)&q=(1_3)(7_3)`

[3]Classical critical event of type b at height ϵ_F is at vertical border connecting the left to the right cell. View the example here: `https://abegehr.github.io/frechet/?p=(1_2)(4_5)(7_2)&q=(1_3)(7_3)`

[4]Classical critical event of type c at height ϵ_F is at vertical border connecting the left to the right cell over the middle cell. View the example here: `https://abegehr.github.io/frechet/?p=(4_3)(7_4)(1_4)(4_3)&q=(1_3)(7_3)`

3.5 Computing the Classical Fréchet Distance

We now have the tools to compute the classical Fréchet distance. To compute the classical Fréchet distance, we can apply *Algorithm 2* of Alt and Godau's paper.[1] The algorithm first determines all classical critical events of type a, b, and c, including their critical ϵ and then using a binary search and the decision algorithm, finds the smallest critical ϵ for which the height function δ is traversable.

4 Lexicographic Fréchet Matchings

In his 2014 paper "Lexicographic Fréchet Matchings", Rote extended upon the findings from Alt and Godau in their paper "Computing the Fréchet distance between two polygonal curves" to establish an algorithm that produces a lexicographic Fréchet matching.

Taking a lexicographic approach roughly means that we want to minimize the time T(s) during which the height exceeds a threshold s.[2] We want to spend as little time as possible at the current height and want to descend as quickly as possible.

4.1 Steepest Descent

To achieve a lexicographically optimized traversal, Rote makes use of the steepest descent. Therefore he developed a steepest descent algorithm.

Because we chose our input to be two paths consisting of straight line segments, the level set in a cell consists of concentric ellipses. Rote defines two lines for every cell, l and l':

(l) The points where the ellipses have vertical tangents and δ has a linear horizontal gradient lie on line l through the common center.[2]

(l') The points where the ellipses have horizontal tangents and δ has a linear vertical gradient lie on line l through the common center.[2]

Depending on whether we are descending from a point on one of the lines or from in between one of the quadrants they span up, the direction of the steepest descent is affected. Consult Rote's paper "Lexicographic Fréchet Matchings"[2] (Figure 5) for more detail. It is important to note at this point, that the steepest descent for a point is unique.

4.2 New Lexicographic Type of Critical Event

Rote also identifies a new lexicographic type of critical event, which does not influence the global Fréchet distance ϵ_F, but is essential for achieving a lexicographically correct traversal.

The new lexicographic type of critical event occurs while descending on a steepest descent path. It occurs when we would horizontally or vertically surpass a critical opening of the height at which we currently are.

Figure 3 shows a minimal example of these new lexicographic types of critical events.

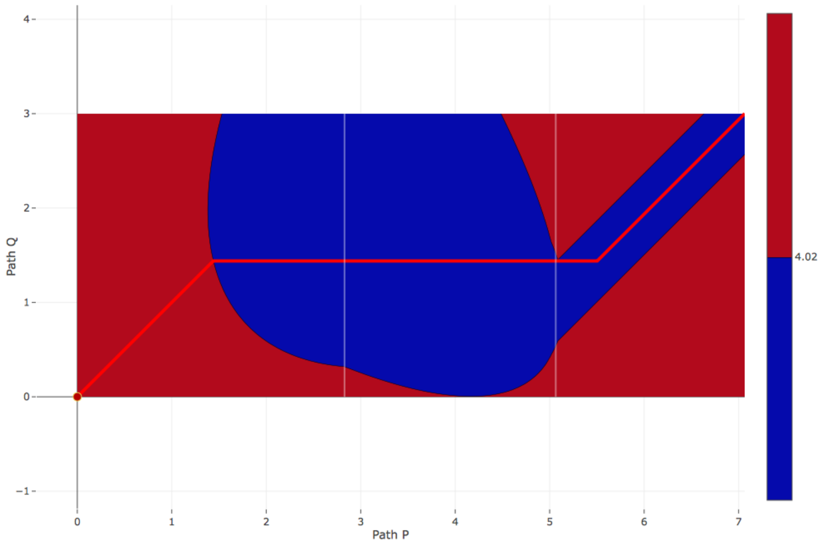


Figure 3: Example of new lexicographic type of critical event[5]

4.3 Lexicographic Fréchet Matching Algorithm

We now have the tools needed to understand Rote's algorithm for lexicographic Fréchet matchings[2] under *general inputs* (not yet regarding degenerate inputs, as described in Section 5). In contrary to the algorithm for computing the classical Fréchet distance, the algorithm for lexicographic Fréchet matching utilizes recursion to allow for a locally correct lexicographic traversal. The algorithm as described here is aligned to the implementation of the online visualization.

[5]New lexicographic critical event of type a below height ϵ_F connecting the middle of the left cell to the vertical border between the middle and the right cell. View the example here: `https://abegehr.github.io/frechet/?p=(3_2)(5_4)(3_3)(5_3)&q=(2_7)(5_7)`

Algorithm 1 Requirements for Lexicographic Fréchet Matching Algorithm

function DecisionAlgorithm(δ, A, B, ϵ)

▷ Implementation of the decision algorithm, that decides if δ is traversable from A to B with ϵ.

  return $true$ or $false$

function SteepestDescent($\delta, A, B, \epsilon_A, \epsilon_B$)

▷ Attempts a steepest descent as far as possible for the higher of A and B, or both if they reside at equal height. Attempt the steepest descent to A' and B' until A' or B' reach a cell border or a line l or l', they would pass each other vertically or horizontally, or they meet each other. Descend to an equal height if possible.

  return $A', B', \epsilon_{A'}, \epsilon_{B'}$

function ReverseDescent($\delta, A, B, \epsilon_A, \epsilon_B$)

▷ Reverses steepest descent until a classical critical event of type b or c, or a new lexicographic critical event is found. Also returns the critical event c.

  return $A', B', \epsilon_{A'}, \epsilon_{B'}, c$

function ClassicalFréchet($\delta, A, B, \epsilon_A, \epsilon_B$)

▷ Using a binary search and the decision algorithm, this function finds the lowest traversable classical critical event that is needed to traverse δ from A to B. This is the same as the algorithm for the classical Fréchet distance.

  return ϵ_F, c

Algorithm 2 Lexicographic Fréchet Matching Algorithm

function TraverseRecursive($\delta, A, B, \epsilon_A, \epsilon_B$)
  if $A = B$ **or** $B.x \leq A.x$ **or** $B.y \leq A.y$ **then**
    ▷ Traversal done. Connect A and B.
    return Connect(A, B)
  $\epsilon \leftarrow \max(\epsilon_A, \epsilon_B)$
  if DecisionAlgorithm(δ, A, B, ϵ) **then**
    $A', B', \epsilon_{A'}, \epsilon_{B'} \leftarrow$ SteepestDescent($\delta, A, B, \epsilon_A, \epsilon_B$)
    $\epsilon' \leftarrow \max(\epsilon_{A'}, \epsilon_{B'})$
    if DecisionAlgorithm($\delta, A', B', \epsilon'$) **then**
      ▷ Traversable after steepest descent.
      return TraverseRecursive($\delta, A', B', \epsilon_{A'}, \epsilon_{B'}$)
    else ▷ Not traversable. The descent passes a critical event.
      while $A \neq A'$ **or** $B \neq B'$ **do**
        $A', B', \epsilon_{A'}, \epsilon_{B'}, c \leftarrow$ ReverseDescent($\delta, A', B', \epsilon_{A'}, \epsilon_{B'}$)
        $\epsilon' \leftarrow \max(\epsilon_{A'}, \epsilon_{B'})$
        if DecisionAlgorithm($\delta, A', B', \epsilon'$) **then**
          ▷ Traversable with the critical event c.
          return TraverseRecursive($\delta, A', c.B, \epsilon_{A'}, \epsilon_c$) + c + TraverseRecursive($\delta, c.A, B', \epsilon_c, \epsilon_{B'}$)
  else ▷ A critical event with $\epsilon_c > \epsilon$ is needed to traverse.
    $\epsilon_F, c \leftarrow$ ClassicalFréchet($\delta, A, B, \epsilon_A, \epsilon_B$)
    return TraverseRecursive($\delta, A', c.B, \epsilon_{A'}, \epsilon_c$) + c + TraverseRecursive($\delta, c.A, B', \epsilon_c, \epsilon_{B'}$)

5 Lexicographic Fréchet Distance with Degenerate Inputs

5.1 Problem

As Rote describes in Section 7 of his paper "Lexicographic Fréchet Matchings" and as we hinted to in Section 4.3 of this paper, multiple critical events can occur for the same ϵ.

Multiple critical events for the same ϵ turn out to be problematic, because as can be seen in Algorithm 2, we assume the functions defined in Algorithm 1 return only one critical event, which will be traversed.

This section aims to answer the question of how to handle multiple critical events with the same critical ϵ by visualizing the problem and postulating an algorithm that decides which combination of critical events result in a lexicographic traversal. The basis of this section is Section 7 of Rote's paper "Lexicographic Fréchet Matchings"[2].

5.2 Examples

In this subsection, four examples will be discussed, for which multiple critical events for the same critical ϵ require a decision on which of them should be traversed to result in a lexicographic traversal.

5.2.1 Minimal Example: Traverse Both

ces_min_1.png

Figure 4

Figure 4 shows a minimal example of multiple critical events for the same ϵ. It can be seen easily

5.2.2 Minimal Example: Traverse One

5.2.3 Minimal Example: Non Trivial

https://abegehr.github.io/frechet/?p=$(0_0)(0_5)(2_2)q = (-1_1)(4_1)(-1_1)$

5.2.4 More Complex Example

Figure **??** shows a more complex example of multiple critical events for the same ϵ.

5.3 Postulate Algorithm

Postulate algorithm for choosing a critical event from several with equal epsilon: (visual examples!)

Algorithm:

1. For every critical event compute reciprocal of derivates descents and store as critical event's steepness.
2. From set of critical events, generate all possible sequences (ascending monotone).
3. For all sequences sum steepness of their critical events and store as sequence rank.
4. Starting with the smallest-rank sequence, decide if the sequence of critical events is traversable without passing critical events excluded in the sequence.
5. If decision is negative, repeat 4.
6. If decision is positive, repeat 4 for all sequences of equal rank.
7. If multiple sequences of equal rank are found to be valid, traverse all, and decide by comparing.
8. The steepness of the steepest decent.
9. The steepness of critical event that is reached.
10. And choose the path with the steepest summed recents of all paths.
11. If multiple paths with same hight profile are found, return all. Otherwise return the lexicographic optimum.
12.

5.4 Traversing critical events

Assume we are traversing a heat map generated by two curves. We arrive at a point where we have to decide which critical events need to be traversed for a lexicographic solution. The decision algorithm[1] can be applied for the ϵ of each possible critical event. We pick the smallest ϵ for which the decision algorithm returns a positive decision. We call it ϵ_0. Now two options are feasible:

(A) There is one critical event at height ϵ_0. $|C_{\epsilon_0}| = 1$

(B) There are multiple critical events at height ϵ_0. $|C_{\epsilon_0}| > 1$

In case option A holds true, this critical event will need to be traversed. The traversal-problem is divided into two sub problems, which are both examined recursively and independently.

In case option B holds true, we need to decide which of the critical events C_{ϵ_0} need to be traversed and which are not needed for an optimal solution. We will consider all subsets C'_{ϵ_0} of C_{ϵ_0} with size $n > 0$.

5.4.1 Traversing multiple critical events with same ϵ

For each subset of critical events at ϵ_0, $C'_{\epsilon_0} \subseteq C_{\epsilon_0}$ we compute if the critical events can be traversed monotonic. This can be done easily by applying the following procedure. For each critical event $C'_{\epsilon_0}(i)$, test if the start- and endpoints of all other critical events in C'_{ϵ_0} are either lower-left of the starting point of $C'_{\epsilon_0}(i)$ or upper-right of the ending point of $C'_{\epsilon_0}(i)$.

See the a pseudocode for the function here. The function will return $True$ if the critical events C'_{ϵ_0} are traversable monotonically and $False$ if they are not.

Algorithm 3 Critical Events C'_{ϵ_0} Monotonic Traversable

function $PLeq(p, q)$
  ▷ determines if two points are monotonically traversable
  return ($p_x \leq q_x$ **and** $p_y \leq q_y$)

function MonotonicTraversable(C'_{ϵ_0})
  ▷ determines if a set of critical events C'_{ϵ_0} are monotonically traversable
  for i in len(C'_{ϵ_0}) **do**
    $c_i \leftarrow C'_{\epsilon_0}(i)$
    $c_{i,s} \leftarrow c_i.start$
    $c_{i,e} \leftarrow c_i.end$

    for j in len(C'_{ϵ_0}) **do**
      if $j == i$ **then** *continue* ▷ c_j is skipped

      $c_j \leftarrow C'_{\epsilon_0}(j)$
      $c_{j,s} \leftarrow c_j.start$
      $c_{j,e} \leftarrow c_j.end$

      if $PLeq(c_{j,s}, c_{i,s})$ **and** $PLeq(c_{j,e}, c_{i,s})$ **then**
        continue ▷ c_j is traversable prior to c_i
      if $PLeq(c_{i,e}, c_{j,s})$ **and** $PLeq(c_{i,e}, c_{j,e})$ **then**
        continue ▷ c_j is traversable after c_i

      return $False$ ▷ c_i and c_j are not traversable monotonically
  return $True$ ▷ the critical events C'_{ϵ_0} are monotonically traversable

$MonotonicTraversable$ is applied to all subsets of critical events at ϵ_0, $C'_{\epsilon_0} \subseteq C_{\epsilon_0}$. We will only consider the critical events C'_{ϵ_0}, for which C'_{ϵ_0} is monotonically traversable, *i.e.* $MonotonicTraversable(C'_{\epsilon_0})$ holds True, because other combinations of critical events are not monotonically traversable. We call this monotonically traversable subset $C''_{\epsilon_0} \subseteq C_{\epsilon_0}$.

$$C''_{\epsilon_0} \coloneqq \{C'_{\epsilon_0} \mid MonotonicTraversable(C'_{\epsilon_0})\}$$

5.5 Example of Algorithm visualised (visual example!)

5.6 Analysing postulated algorithm for handling degenerate inputs

5.6.1 Runtime analysis

5.7 Real world application

Can you think of one?

6 Concluding Perspective

Concluding, all is great! :)

References

[1] Helmut Alt and Michael Godau. "Computing the Fréchet distance between two polygonal curves". In: *International Journal of Computational Geometry & Applications* 5.01n02 (1995), pp. 75–91.

[2] Günter Rote. "Lexicographic Fréchet Matchings". In: *30th European Workshop on Computational Geometry*. EuroCG. 2014.