When Lipschitz Walks Your Dog: 
Algorithm Engineering of the 
Discrete Fréchet Distance under Translation

Karl Bringmann, Marvin Künemann, and André Nusser
Teaser

**Trajectory Similarity:**
- Pigeon GPS Trajectories
- Handwritten Characters

**Fréchet Distance:**
- traversal based
- fast in practice

**Fréchet Distance Under Translation:**
- traversal based
- only impractical algorithms (before)
- translation invariant
Teaser

Best algorithm: $O(n^{4.66})$

Conditional lower bound: $n^{4-o(1)}$

All algorithms build $O(n^4)$ arrangement!

Fréchet under translation is 1-Lipschitz in $\tau$!

Use continuous optimization:
- branch & bound!
Teaser

**Take-Home Message:**

- **arrangement-based geometric algorithm**
- **methods from continuous optimization**
- **exact**
- **approximation**
- **fast practical algorithm :)**

SAME-CHARACTERS:

<table>
<thead>
<tr>
<th>Decision time (ms)</th>
<th>distance factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 10^-1</td>
<td>1 - 10^-2</td>
</tr>
<tr>
<td>1 - 10^-2</td>
<td>1 - 10^-3</td>
</tr>
<tr>
<td>1 - 10^-3</td>
<td>1 - 10^-4</td>
</tr>
<tr>
<td>1 - 10^-4</td>
<td>1 - 10^-5</td>
</tr>
<tr>
<td>1 - 10^-5</td>
<td>1 - 10^-6</td>
</tr>
<tr>
<td>1 - 10^-6</td>
<td>1 - 10^-7</td>
</tr>
<tr>
<td>1 - 10^-7</td>
<td>1 - 10^-8</td>
</tr>
<tr>
<td>1 - 10^-8</td>
<td>1 - 10^-9</td>
</tr>
<tr>
<td>1 - 10^-9</td>
<td>1 - 10^-10</td>
</tr>
<tr>
<td>1 - 10^-10</td>
<td>1 - 10^-11</td>
</tr>
</tbody>
</table>

SAME-CHARACTERS:

<table>
<thead>
<tr>
<th>Black-box calls</th>
<th>distance factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^1</td>
<td>1 - 10^-1</td>
</tr>
<tr>
<td>10^2</td>
<td>1 - 10^-2</td>
</tr>
<tr>
<td>10^3</td>
<td>1 - 10^-3</td>
</tr>
<tr>
<td>10^4</td>
<td>1 - 10^-4</td>
</tr>
<tr>
<td>10^5</td>
<td>1 - 10^-5</td>
</tr>
<tr>
<td>10^6</td>
<td>1 - 10^-6</td>
</tr>
<tr>
<td>10^7</td>
<td>1 - 10^-7</td>
</tr>
<tr>
<td>10^8</td>
<td>1 - 10^-8</td>
</tr>
<tr>
<td>10^9</td>
<td>1 - 10^-9</td>
</tr>
<tr>
<td>10^10</td>
<td>1 - 10^-10</td>
</tr>
</tbody>
</table>

**Algorithm Engineering of the Discrete Fréchet Distance under Translation**

Karl Bringmann, Marvin Künnemann, and André Nusser
End of Teaser
Why Trajectory Similarity?

Pigeons’ GPS Trajectories:

Handwritten Character Trajectories:
Discrete Fréchet Distance

Intuition

human

dog
Algorithm Engineering of the Discrete Fréchet Distance under Translation

Discrete Fréchet Distance

Intuition

human

dog
Discrete Fréchet Distance
Intuition

human

dog

Algorithm Engineering of the Discrete Fréchet Distance under Translation

Karl Bringmann, Marvin Künemann, and André Nusser
Intuition

Discrete Fréchet Distance

human

dog
**Discrete Fréchet Distance**

**Intuition**

**Question:** What is the traversal that achieves the shortest leash length?
Discrete Fréchet Distance
Formal Definition

\[ \delta_F(\pi, \sigma) := \min_{f,g \in \mathcal{T}} \max_{t \in [0,1]} \| \pi f(t) - \sigma g(t) \| \]

\( \pi, \sigma = \) polygonal curves of length \( n \)
\( \mathcal{T} = \) set of monotone and surjective functions from \([0, 1]\) to \(\{1, \ldots, n\}\)
Discrete Fréchet Distance under Translation

Definition

**Intuition:** Allow arbitrary translations $\tau \in \mathbb{R}^2$ of curve $\sigma$.

$$\delta_T(\pi, \sigma) := \min_{\tau \in \mathbb{R}^2} \delta_F(\pi, \sigma + \tau)$$
Discrete Fréchet Distance under Translation

Definition

Intuition: Allow arbitrary translations $\tau \in \mathbb{R}^2$ of curve $\sigma$.

\[ \delta_T(\pi, \sigma) := \min_{\tau \in \mathbb{R}^2} \delta_F(\pi, \sigma + \tau) \]
Discrete Fréchet Distance under Translation

Definition

**Intuition:** Allow arbitrary translations $\tau \in \mathbb{R}^2$ of curve $\sigma$.

\[
\delta_T(\pi, \sigma) := \min_{\tau \in \mathbb{R}^2} \delta_F(\pi, \sigma + \tau)
\]
Discrete Fréchet Distance under Translation

**Definition**

**Intuition:** Allow arbitrary translations $\tau \in \mathbb{R}^2$ of curve $\sigma$.

\[
\delta_T(\pi, \sigma) := \min_{\tau \in \mathbb{R}^2} \delta_F(\pi, \sigma + \tau)
\]

**Decision Problem:**
- Given $\pi, \sigma, \delta$
- $\delta_T(\pi, \sigma) \leq \delta$?
**Discrete Fréchet Distance under Translation**

**Definition**

**Intuition:** Allow arbitrary translations $\tau \in \mathbb{R}^2$ of curve $\sigma$.

$$\delta_T(\pi, \sigma) := \min_{\tau \in \mathbb{R}^2} \delta_F(\pi, \sigma + \tau)$$

**Decision Problem:**

- Given $\pi, \sigma, \delta$
- $\delta_T(\pi, \sigma) \leq \delta$?

Focus on this in the talk!
Goal:
Performant implementation computing the discrete Fréchet distance under translation on practical inputs.
Related Work

Theory:

- Discrete Fréchet distance under translation in $\tilde{O}(n^5)$
  [Agarwal, Ben Avraham, Kaplan, Sharir arXiv'15]
- Discrete Fréchet distance under translation in $\tilde{O}(n^{4.66})$
  [Bringmann, Künemann, N. SODA'19]
- SETH based lower bound of $n^{4-o(1)}$ for discrete Fréchet distance under translation
  [Bringmann, Künemann, N. SODA'19]
Related Work

Theory:

- Discrete Fréchet distance under translation in $\tilde{O}(n^5)$
  [Agarwal, Ben Avraham, Kaplan, Sharir arXiv'15]
- Discrete Fréchet distance under translation in $\tilde{O}(n^{4.66})$
  [Bringmann, Künnemann, N. SODA'19]
- SETH based lower bound of $n^{4-o(1)}$ for discrete Fréchet distance under translation
  [Bringmann, Künnemann, N. SODA'19]

Practice:

- GIS Cup on (fixed-translation) Fréchet distance near neighbors search
  [Werner, Oliver; Baldus et al.; Buchin et al.; Dütsch et al. SIGSPATIAL'17]
- State of the art (fixed-translation) Fréchet distance implementation
  [Bringmann, Künnemann, N. SoCG'19]
Approach I: Discrete Algorithms

Arrangement

- Idea: Partition the plane into equivalent regions.
Approach I: Discrete Algorithms

Arrangement

- Idea: Partition the plane into equivalent regions.

\[ \pi, \sigma, \delta, \tau_1, \tau_2 \]
Approach I: Discrete Algorithms

Arrangement

- Idea: Partition the plane into equivalent regions.
Approach I: Discrete Algorithms

Arrangement

• Idea: Partition the plane into equivalent regions.
Approach I: Discrete Algorithms

Arrangement

• Idea: Partition the plane into equivalent regions.
Approach I: Discrete Algorithms

Arrangement

- Idea: Partition the plane into equivalent regions.
Approach I: Discrete Algorithms

Arrangement

- Idea: Partition the plane into equivalent regions.
Approach I: Discrete Algorithms

Arrangement

- Idea: Partition the plane into equivalent regions.
Approach I: Discrete Algorithms

Arrangement

• Idea: Partition the plane into equivalent regions.

Observation: All translations in a cell of the arrangement have the same closeness relation.
Approach I: Discrete Algorithms

Arrangement

- Idea: Partition the plane into equivalent regions.

Observation: All translations in a cell of the arrangement have the same closeness relation.

for each cell, pick some $\tau$ and check $d_F(\pi, \sigma + \tau)$
Approach I: Discrete Algorithms

Arrangement

- Idea: Partition the plane into equivalent regions.

**Observation:** All translations in a cell of the arrangement have the same closeness relation.

For each cell, pick some $\tau$ and check $d_F(\pi, \sigma + \tau)$

$O(n^4)$ complexity
Approach I: Discrete Algorithms

- All known algorithms build an $O(n^4)$ arrangement.
Approach I: Discrete Algorithms

• All known algorithms build an $O(n^4)$ arrangement.
Approach I: Discrete Algorithms

- All known algorithms build an $O(n^4)$ arrangement.
Approach I: Discrete Algorithms

- All known algorithms build an $O(n^4)$ arrangement.
Approach II: Continuous Optimization
Approach II: Continuous Optimization

![Diagram showing Fréchet Distance under Translation]
Observation: Fréchet under Translation is 1-Lipschitz, i.e.,

$$|d_F(\pi, \sigma + \tau) - d_F(\pi, \sigma + \tau')| \leq \|\tau - \tau'|$$
Approach II: Continuous Optimization

Lipschitz Optimization

Approach:
- branch & bound

For each box:
Approach II: Continuous Optimization

Lipschitz Optimization

Approach:

- branch & bound

For each box:

- if $d_F(\pi, \sigma + \square) \leq \delta$
  - return LESS
Approach II: Continuous Optimization

Lipschitz Optimization

**Approach:**
- branch & bound

**For each box:**
- if \( d_F(\pi, \sigma + \tau B) \leq \delta \)  
  - return LESS
- if \( d_F(\pi, \sigma + \tau B) > \delta + \)  
  - skip box

\[Q \leftarrow \text{Fifo}(\text{initial search box})\]
while \( Q \neq \emptyset \) do
\[B \leftarrow \text{extract front of search box queue} \]
if \( \text{FréchetDistance}(\pi, \sigma + \tau B) > \delta + d_B/2 \) then
  skip \( B \)
endif
if \( \text{FréchetDistance}(\pi, \sigma + \tau B) \leq \delta \) then
  return YES
endif
halve \( B \) along longest edge and push resulting child boxes to \( Q \)
end while
return NO
Approach II: Continuous Optimization

Lipschitz Optimization

Approach:

- branch & bound

For each box:

- if $d_F(\pi, \sigma + ) \leq \delta$
  - return LESS
- if $d_F(\pi, \sigma + ) > \delta +$
  - skip box
- if both fail: split
Approach II: Continuous Optimization

Issues

• In general, only approximate decisions possible.
Approach II: Continuous Optimization

Issues

• In general, only approximate decisions possible.
• Locally highly non-convex:
Approach II: Continuous Optimization

Issues

- In general, only approximate decisions possible.
- Locally highly non-convex:
Approach II: Continuous Optimization

Issues

• In general, only approximate decisions possible.
• Locally highly non-convex:
Core Idea

Combine Both Approaches!
Core Idea

Combine Both Approaches!

1) Use Lipschitz optimization to identify important regions
Core Idea

Combine Both Approaches!

1) Use Lipschitz optimization to identify important regions
2) Use arrangement algorithm inside these regions
Contribution I: Exact Decider
Contribution I: Exact Decider

Approach:
- augment branch & bound approach
- for each box:
  - estimate arrangement size
- if it is small: build arrangement

exact decision!
Contribution I: Exact Decider

Approach:
- augment branch & bound approach
- for each box:
  - estimate arrangement size
- if it is small: build arrangement

exact decision!
Contribution I: Exact Decider

Approach:

- augment branch & bound approach
- for each box:
  - estimate arrangement size
- if it is small: build arrangement

exact decision!
Karl Bringmann, Marvin Künemann, and André Nusser

Algorithm Engineering of the Discrete Fréchet Distance under Translation

Contribution I: Exact Decider

Approach:

- augment branch & bound approach
- for each box:
  - estimate arrangement size
- if it is small: build arrangement

exact decision!
Contribution I: Exact Decider

Approach:

• augment branch & bound approach
• for each box:
  – estimate arrangement size
• if it is small: build arrangement

exact decision!
Contribution I: Exact Decider

Approach:
- augment branch & bound approach
- for each box:
  - estimate arrangement size
- if it is small: build arrangement

exact decision!
Contribution I: Exact Decider

Approach:
- augment branch & bound approach
- for each box:
  - estimate arrangement size
- if it is small: build arrangement

exact decision!

Issue: When to build the arrangement?

Main Ingredients:
1. Arrangement size estimation
2. Threshold parameter
Contribution I: Exact Decider

Approach:
- augment branch & bound approach
- for each box:
  - estimate arrangement size
- if it is small: build arrangement

Issue: When to build the arrangement?

Main Ingredients:
1. Arrangement size estimation
2. Threshold parameter

modified kd-tree
Contribution I: Exact Decider

Approach:

- augment branch & bound approach
- for each box:
  - estimate arrangement size
- if it is small: build arrangement

exact decision!

Issue: When to build the arrangement?

Main Ingredients:

1. Arrangement size estimation
2. Threshold parameter

empirically choose modified kd-tree
Contribution I: Exact Decider

Implementation Details

- Adaption of (fixed-translation) [SoCG’19] implementation to discrete case
- Lazy translation
- Parameter choice for arrangement size estimation
Contribution II: From Decider to Value Computation
Contribution II: From Decider to Value Computation

Approaches

Epsilon-approximate Set:

$O(\varepsilon)$
Contribution II: From Decider to Value Computation

Approaches

Epsilon-approximate Set:

Binary Search via Decision Problem:

- Binary search over $\delta$ using decider
Contribution II: From Decider to Value Computation

Approaches

Epsilon-approximate Set:
\[ O(\epsilon) \]

Lipschitz-only Optimization:
- Use plain Lipschitz optimization

Binary Search via Decision Problem:
- Binary search over \( \delta \) using decider
Contribution II: From Decider to Value Computation

Approaches

**Epsilon-approximate Set:**
- $O(\epsilon)$
- Next slide

**Binary Search via Decision Problem:**
- Binary search over $\delta$ using decider

**Lipschitz-only Optimization:**
- Use plain Lipschitz optimization

**Lipschitz-meets-Fréchet:**
- Next slide
Contribution II: From Decider to Value Computation

Lipschitz-meets-Fréchet

Approach:
1. Maintain local lower bound
2. Maintain global upper bound
3. Arrangement size estimation

For each box:

adapt!
Contribution II: From Decider to Value Computation

Lipschitz-meets-Fréchet

**Approach:**
1. Maintain local lower bound
2. Maintain global upper bound
3. Arrangement size estimation

**For each box:**
- update *global* upper bound:
  - $\min\{ub, d_F(\pi, \sigma + \square)\}$
Contribution II: From Decider to Value Computation

Lipschitz-meets-Fréchet

Approach:
1. Maintain local lower bound
2. Maintain global upper bound
3. Arrangement size estimation

For each box:
- update global upper bound:
  - \( \min\{ub, d_F(\pi, \sigma+ )\} \)
- update local lower bound:
  - \( \max\{lb, d_F(\pi, \sigma+ ) - \} \)
Contribution II: From Decider to Value Computation

Approach:
1. Maintain local lower bound
2. Maintain global upper bound
3. Arrangement size estimation

For each box:
- update global upper bound:
  - \( \min\{\text{ub}, d_F(\pi, \sigma+)\} \)
- update local lower bound:
  - \( \max\{\text{lb}, d_F(\pi, \sigma+)-\} \)
- if \( \text{ub} > \text{lb} + \epsilon \): split
Contribution II: From Decider to Value Computation

Lipschitz-meets-Fréchet

Approach:
1. Maintain local lower bound
2. Maintain global upper bound
3. Arrangement size estimation

For each box:
- Update global upper bound:
  \[ \min\{ub, d_F(\pi, \sigma+ )\} \]
- Update local lower bound:
  \[ \max\{lb, d_F(\pi, \sigma+ ) - \} \]
- If \( ub > lb + \epsilon \): split

Binary search over arrangement!
Contribution II: From Decider to Value Computation
Lipschitz-meets-Fréchet: Implementation Details

- Arrangement size estimation
- Initial estimates
- Priority queue on lower bound $\rightarrow$ no regret strategy!
## Experiments

### Data Sets

<table>
<thead>
<tr>
<th>Data set</th>
<th>Type</th>
<th>#Curves</th>
<th>Mean #vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sigspatial</strong></td>
<td>synthetic GPS-like</td>
<td>20199</td>
<td>247.8</td>
</tr>
<tr>
<td><strong>Characters</strong></td>
<td>20 handwritten chars</td>
<td>2858</td>
<td>120.9 (142.9 per character)</td>
</tr>
</tbody>
</table>
Experiments

Running Times

**SAME-CHARACTERS:**

![Graph showing decision time vs distance factor](image-url)
Experiments

Running Times

SIGSPATIAL:
Experiments
Running Times

SIGSPATIAL:

• Hard instances are distances slightly less than actual distance
• Running times of hard instances in the order of 100ms
Experiments

Black box calls vs. arrangement size

**SAME-CHARACTERS:**

![Graph showing black box calls vs. distance factor](image-url)
Experiments

Black box calls vs. arrangement size

SIGSPATIAL:
Experiments

Black box calls vs. arrangement size

SIGSPATIAL:

- several orders of magnitudes less calls to black-box decider
# Experiments

## Value Computation Times

<table>
<thead>
<tr>
<th>Approach</th>
<th>Time</th>
<th>Black-Box Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Computation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Times</td>
<td></td>
</tr>
<tr>
<td>LMF</td>
<td>148,032 ms</td>
<td>13,323,232</td>
</tr>
<tr>
<td></td>
<td>(141.0 ms per instance)</td>
<td>(12,688.8 per instance)</td>
</tr>
<tr>
<td>Binary Search</td>
<td>536,853 ms</td>
<td>45,909,628</td>
</tr>
<tr>
<td></td>
<td>(511.3 ms per instance)</td>
<td>(43,723.5 per instance)</td>
</tr>
<tr>
<td>Lipschitz-only</td>
<td>4,204,521 ms</td>
<td>820,468,224</td>
</tr>
<tr>
<td></td>
<td>(4,004.3 ms per instance)</td>
<td>(781,398.3 per instance)</td>
</tr>
</tbody>
</table>
Experiments

Binary Search vs. LMF
Experiments

Binary Search vs. LMF

- LMF better on hard instances
Summary

arrangement-based geometric algorithm

methods from continuous optimization

expensive

approximation

exact
Summary

arrangement-based geometric algorithm

exact

methods from continuous optimization

approximation

fast practical algorithm :)
Summary

Future Directions:

• Apply approach to other problems
• Find optimal point of building the arrangement

arrangement-based geometric algorithm

methods from continuous optimization

expensive

approximation

fast practical algorithm :)
Summary

- arrangement-based geometric algorithm
- methods from continuous optimization
- fast practical algorithm :)