ULB UNIVERSITÉ LIBRE DE BRUXELLES

Trajectory Segmentation

Maxime Hauwaert

Université Libre de Bruxelles

30^{th} August of 2022

ULB FACULTÉ DES SCIENCES

Outline

1. Introduction
2. Trajectory Segmentation
3. Implementation
4. Experimentation
5. Conclusion

Context

- Number of GPS powered devices increases
- Large quantity of data generated
- Efficient databases which support GPS data required
- MobilityDB, a PostgreSQL extension

Trajectory Segmentation Definition

Segmentation

A segmentation of a trajectory τ is a collection of subtrajectories which, when put end-to-end, reforms the original trajectory

$$S(\tau) = [\tau_1, \tau_2, ..., \tau_n]$$

Goals:

- Meaningful subtrajectories, domain dependent
- Homogenous points within each subtrajectory
- Balance between the number of segments and their length

Motivations

- Biology:
  - Study the characteristics of the migrations of certain animals
- Traffic dynamics:
  - Study traffic to improve the traffic flow
- Path planning:
  - Determine the best transportation mode to take
- Tourism:
  - Identify the most visited places by tourists

Outline

1. Introduction

2. Trajectory Segmentation
   Stable Criteria Segmentation
   Sliding Window Segmentation

3. Implementation

4. Experimentation

5. Conclusion

Types of Algorithms

Criteria based

- Characteristics of subtrajectories explicitly stated
- Great knowledge required
- Lots of possibilities

Model based

- Hidden underlying criteria
- Only a few parameters to set
- Easier to use

Stable Criteria Segmentation

Key points:

- Stable criteria
- Combinations
- Trajectory → Compressed start-stop matrix → Segmentation tree → Segmentation
- Parameter:
  - Propositional formula
- Complexity $O(n \log n)$
- Competitors: smaller number of criteria supported, $O(n^2)$

Types of Criteria

Decreasing monotone criteria

If the criterion is met for a trajectory, it is also met for every subtrajectory

Example: speed range

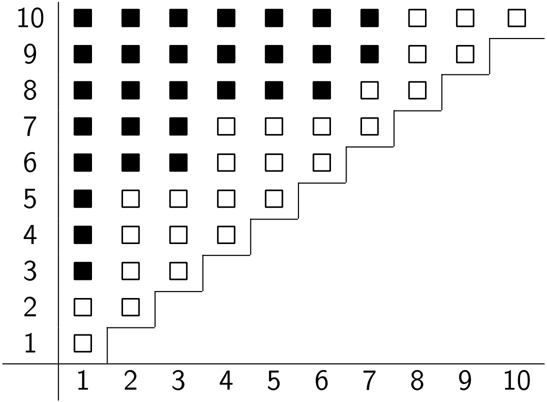
Increasing monotone criteria

If the criterion is met for a trajectory, it is also met for every supertrajectory

Example: time

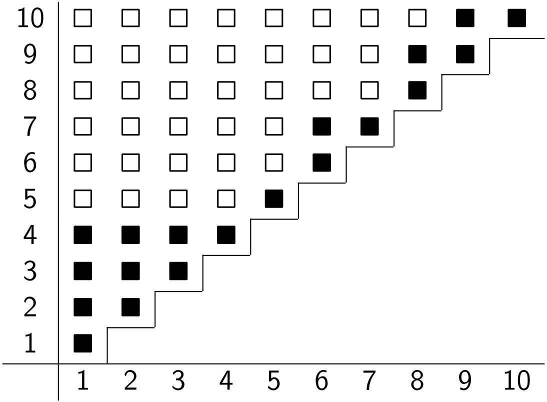
⚠ An increasing monotone criterion is useless on its own, but can be combined with a decreasing monotone criterion

Start-Stop Matrix



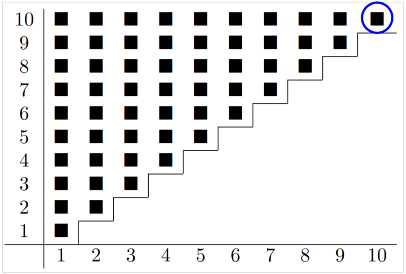
Example of a start-stop matrix obtained with a decreasing monotone criterion

Start-Stop Matrix



Example of a start-stop matrix obtained with an increasing monotone criterion

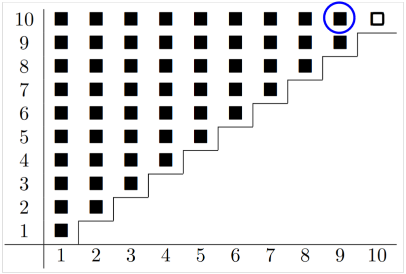
Compressed Start-Stop Matrix Generation



For decreasing monotone criteria:

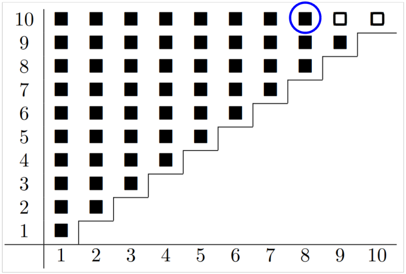
- Start by placing a pointer at the top right of the matrix

Compressed Start-Stop Matrix Generation



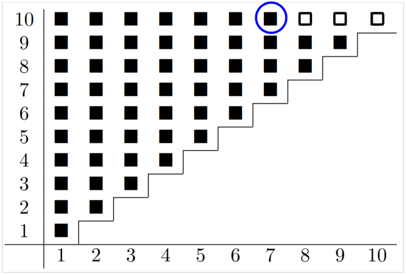
- While the criterion is met for $\tau_{j,i}$ → set i,j to true and decrement j

Compressed Start-Stop Matrix Generation



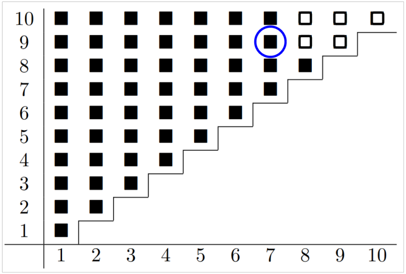
- While the criterion is met for $\tau_{j,i}$ → set i,j to true and decrement j

Compressed Start-Stop Matrix Generation



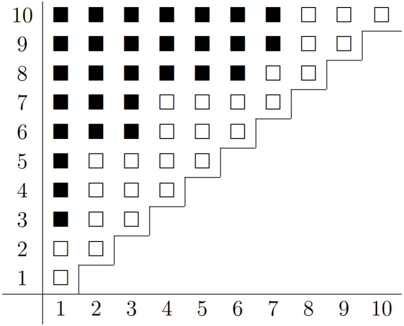
- If the criterion is not met for $\tau_{j,i}$ $\rightarrow$ decrement i, set every element at the right of i,j to true and continue the execution

Compressed Start-Stop Matrix Generation



- If the criterion is not met for $\tau_{j,i}$ → decrement i, set every element at the right of i,j to true and continue the execution

Compressed Start-Stop Matrix Generation



Resulting start stop matrix

Compressed Start-Stop Matrix Generation

For increasing monotone criteria:

Decreasing $\leftrightarrow$ Increasing

- Negation of increasing criterion = decreasing criterion
- Negation of decreasing criterion = increasing criterion

$\rightarrow$ Negate the criteria and then negating the resulting start-stop matrix

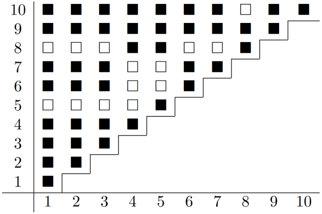
Run length encoding $\rightarrow O(n \log n)$ space complexity

Segmentation Tree Generation

Optimal segmentation of $\tau_{0,i}$:

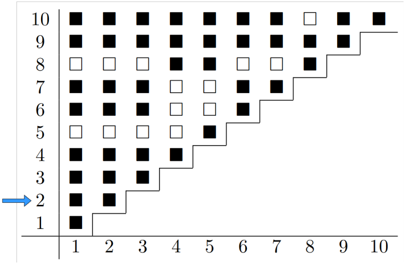
- Either $\tau_{0,i}$
- Or optimal segmentation of $\tau_{0,j} + \tau_{j,i}$

Segmentation Tree Generation



- Iteration on each row of the start-stop matrix
- Keeping a list of tuples:
  - index: the index of a trajectory point
  - count: the number of subtrajectories required to get to the current point
  - last: the last segmentation point
- Adding tuple (index $= 1$, count $= 0$, last $= N/A$)
- Starting at row 2

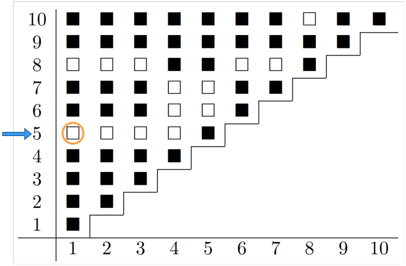
Segmentation Tree Generation



$i = 2$

- No true value at row 2, thus 2 cannot be a segmentation point

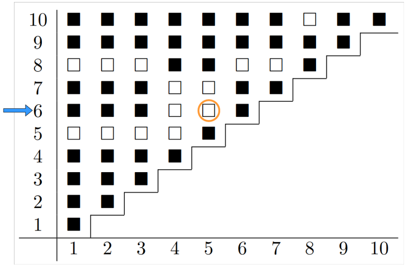
Segmentation Tree Generation



$i = 5$

- There are some true values, thus 5 might be a segmentation point
- Point 5 can be reached from point 1,2,3 and 4
- Select the element with the minimum count.
- Add (index = 5, count = 1, last = 1) to the list

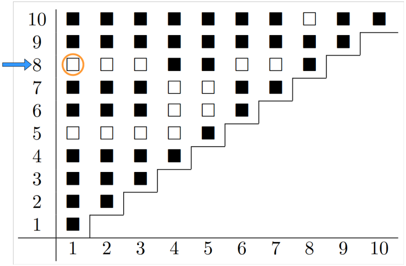
Segmentation Tree Generation



$i = 6$

- Point 6 can be reached from point 4 and 5
- Select the minimum count
- Add (index = 6, count = 2, last = 5) to the list

Segmentation Tree Generation

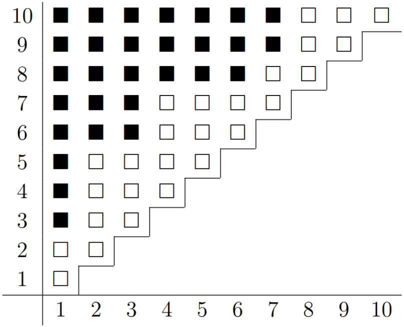


$i = 8$

To speed up:

- Store the tuples in a search binary tree sorted on the index
- Each node points the node with the lowest count in its subtree
- For each block of consecutive true values, search only the first and last index

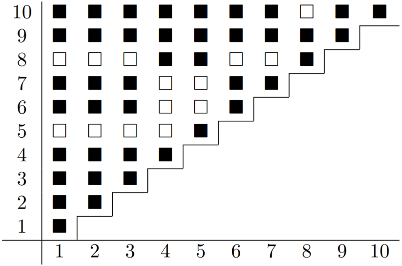
Segmentation Tree Generation



To retrieve the segmentation:

- Take the tuple with the highest index
- Loop on the `last` field until the end is reached
- Outputting the index at each iteration

Segmentation Tree Generation



$\{(5,1,1),\ (6,2,5),\ (7,2,5),\ (8,1,1),\ (10,2,8)\}$

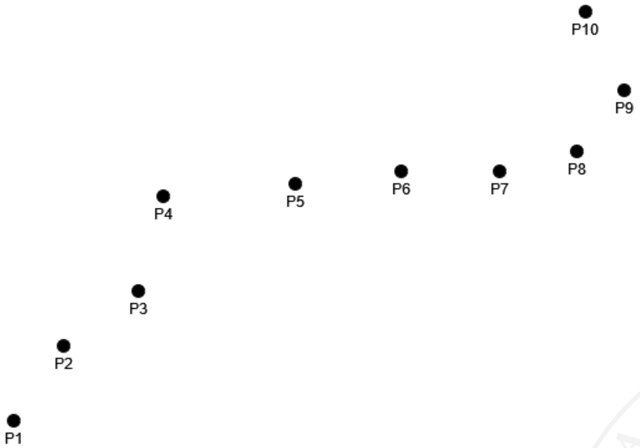
$$\text{Segmentation}(\tau) = [\tau_{1,8},\ \tau_{8,10}]$$

Sliding Window Segmentation

Key points:

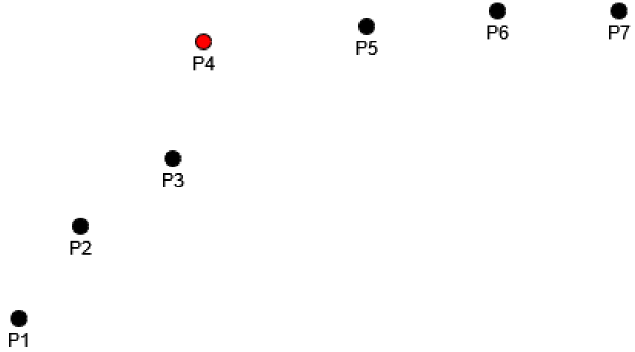
- Sliding window
- Interpolation
- Trajectory → Error signal → Segmentation
- Parameters:
  - Window length
  - Kernel
  - Threshold / Percentile
- Complexity $O(n)$ / $O(n \log n)$
- Competitors: smaller harmonic mean of purity and coverage metrics

Error Signal Generation



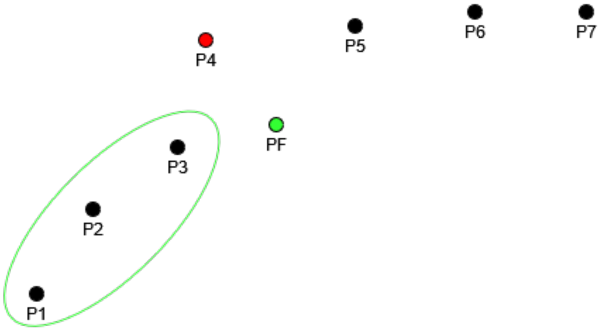
Base trajectory

Error Signal Generation



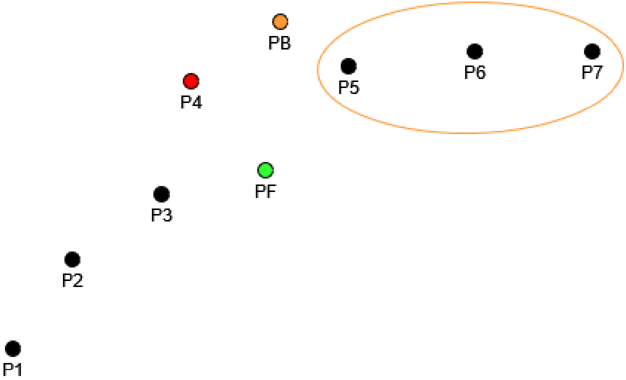
Window creation (length 7)

Error Signal Generation



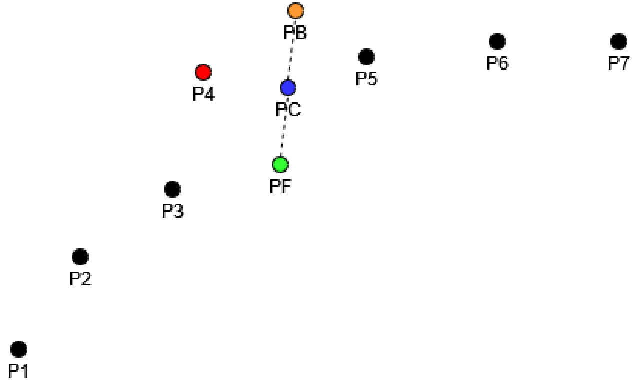
Forward extrapolation

Error Signal Generation



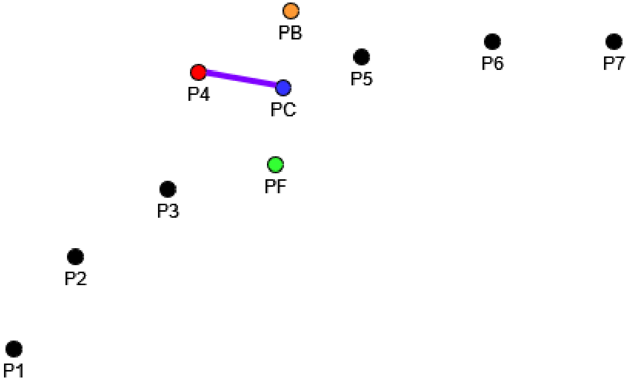
Backward extrapolation

Error Signal Generation



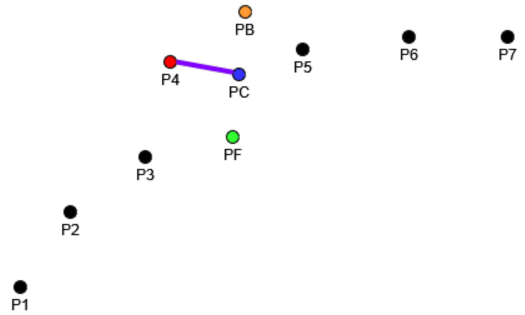
Middle point computation

Error Signal Generation



Error value computation

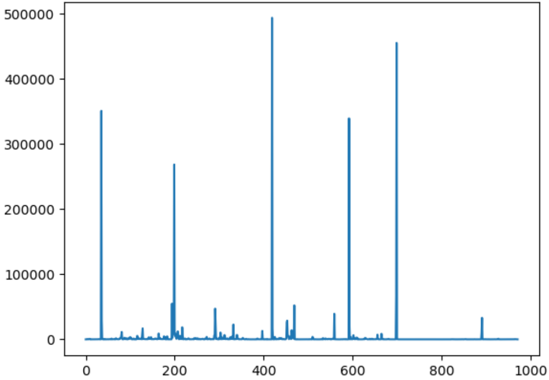
Error Signal Generation



Error value computation

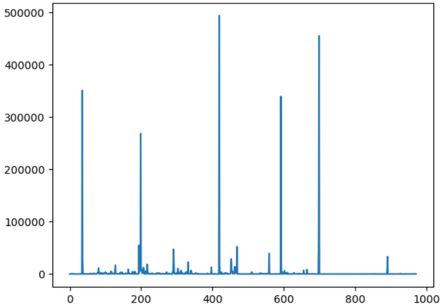
- First and last $\frac{window\_length}{2}$ points impossible to extrapolate $\rightarrow$ error value $= 0$

Segmentation



Error signal

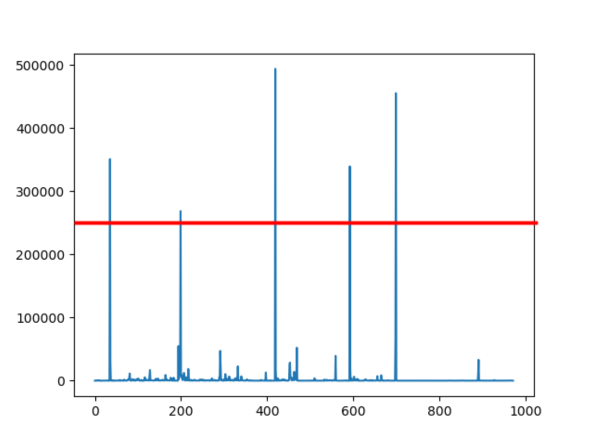
Segmentation



Error signal

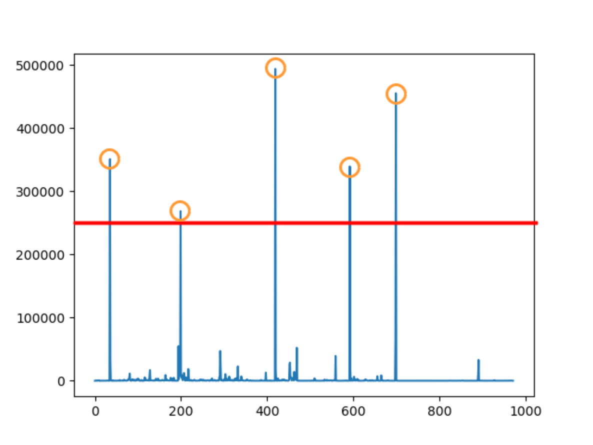
For the percentile variant: set threshold t such that $x\%$ error values $< t$

Segmentation



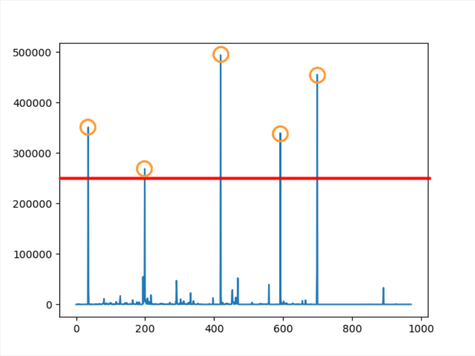
Threshold $= 250000$

Segmentation



Segmentation points

Segmentation



Segmentation points

$$\text{Segmentation}(\tau) = [\tau_{0,25}, \tau_{25,200}, \tau_{200,410}, \tau_{410,590}, \tau_{590,700}, \tau_{700,980}]$$

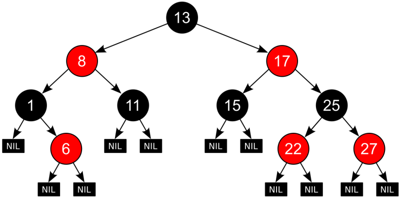
Outline

1. Introduction
2. Trajectory Segmentation
3. Implementation
4. Experimentation
5. Conclusion

Helper algorithms

Red-Black trees

- Binary search tree
- Complexity $\log n$ search, insert, delete
- Augmentation



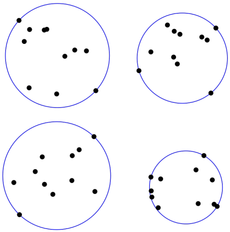
[a]

[a] By Nomen4Omen - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=83966511

Helper algorithms

Welzl's algorithm

- Smallest enclosing disk
- Randomized procedure
- Linear expected time complexity



[a]

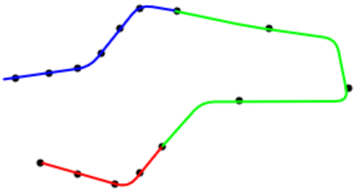
[a] By Claudio Rocchini - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=12562519

Stable Criteria Segmentation

Speed range criterion

For each subtrajectory, the speed of the object should not vary more than the specified amount

Before

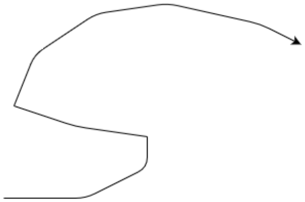


After

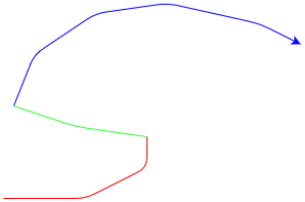
Stable Criteria Segmentation

Angle criterion

Within each subtrajectory, the direction the object is heading does not change by more than the specified angle between two consecutive points



Before

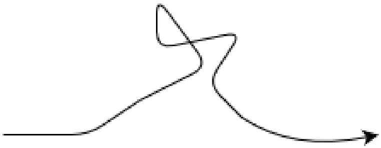


After

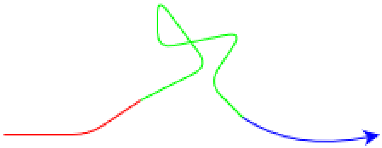
Stable Criteria Segmentation

Disk criterion

For each subtrajectory, there exists a disk of the specified perimeter which covers all of their points



Before



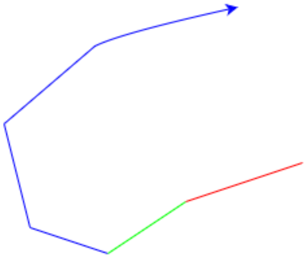
After

Stable Criteria Segmentation

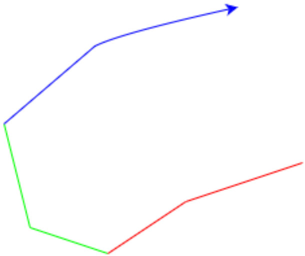
Time criterion

The subtrajectories should last at least the specified amount of time

⚠ Needs to be combined with a decreasing monotone criterion



Before



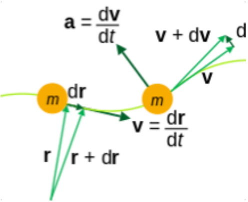
After

Sliding Window Segmentation

Kinematic:

$$x_i = \frac{1}{2} a_x(2)\ (t_i - t_2)^2 + v_x(2)\ (t_i - t_2) + x_2$$

$$v_x(a) = \frac{x_a - x_{a-1}}{t_a - t_{a-1}},\ a_x(a) = \frac{v_x(a) - v_x(a-1)}{t_a - t_{a-1}}$$



[a]

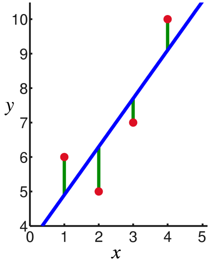
[a] By Maschen - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=21275429

Sliding Window Segmentation

Linear regression:

- Ordinary least squares

$$y = mx + p, \;\; m = \frac{\Sigma((x-\overline{x})(y-\overline{y}))}{\Sigma((x-\overline{x})^2)}, \;\; p = \overline{y} - m\overline{x}$$



[a]

---

[a] By Oleg Alexandrov - self-made with MATLAB, Public Domain, https://commons.wikimedia.org/w/index.php?curid=4099808

Outline

1. Introduction
2. Trajectory Segmentation
3. Implementation
4. Experimentation
5. Conclusion

Scalability

Queries:

- Stable criteria segmentation:

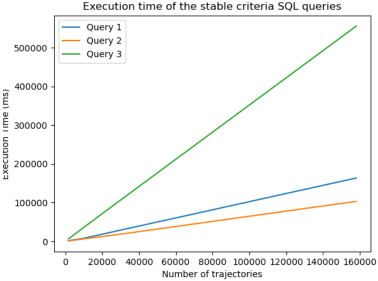
```
1) select segment_by_stable_criteria(trip, '(s[20])') from trips;
2) select segment_by_stable_criteria(trip, '(a[90])') from trips;
3) select segment_by_stable_criteria(trip, '(d[100])') from trips;
```

- Sliding window segmentation:

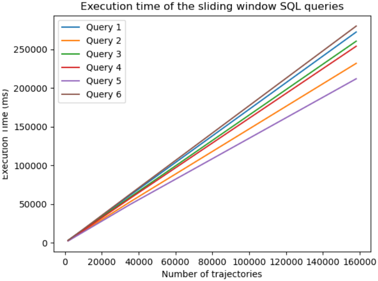
```
1) select segment_by_sws_t(trip, 400000, 7, 'K') from trips;
2) select segment_by_sws_t(trip, 400000, 5, 'L') from trips;
3) select segment_by_sws_p(trip, 85, 7, 'K') from trips;
4) select segment_by_sws_p(trip, 95, 7, 'K') from trips;
5) select segment_by_sws_p(trip, 95, 5, 'L') from trips;
6) select segment_by_sws_p(trip, 95, 9, 'L') from trips;
```

Scalability

Dataset: MobilityDB Berlin mod with four scale factors (0.005,0.05,0.2,1)
Hardware: Windows 11 64 bits, Intel®Core™ i7-7700HQ at 2.8 GHz, 16 GB RAM
Software: PostgreSQL 13, PostGIS 3 and MobilityDB develop



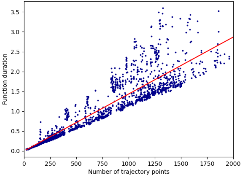
Stable Criteria Segmentation



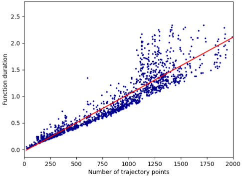
Sliding Window Segmentation

Stable criteria segmentation:

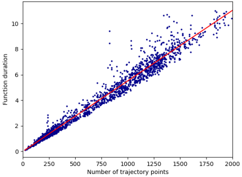
Function duration(y) in relation to the number of trajectory points(x)



Speed range



Angle

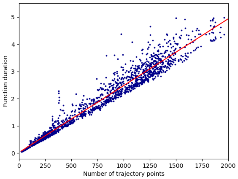


Disk

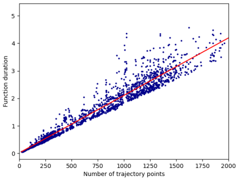
Complexity

Sliding window segmentation:

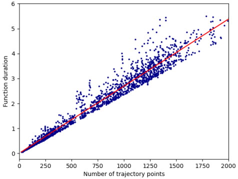
Function duration(y) in relation to the number of trajectory points(x)



Percentile kinematic

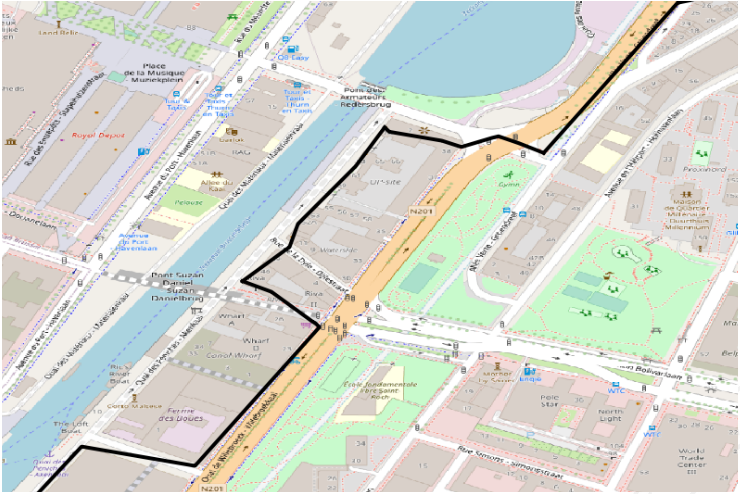


Percentile linear



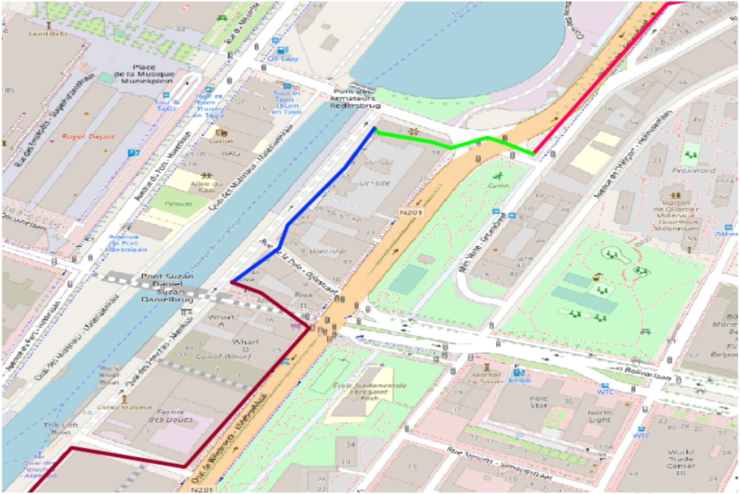
Threshold kinematic

Example



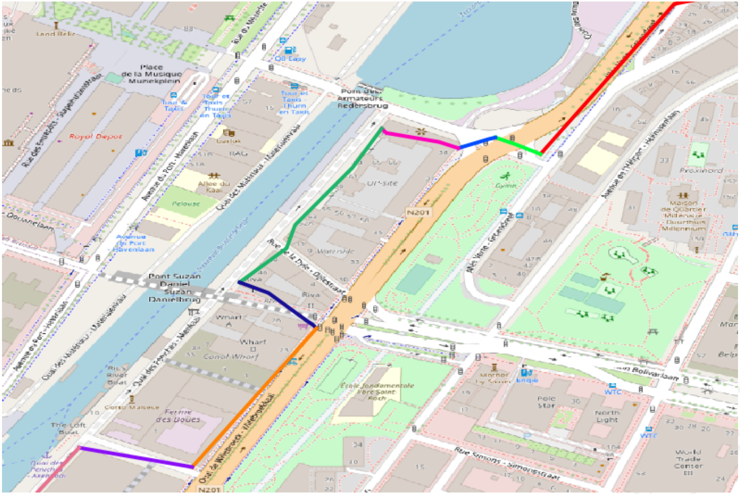
Base trajectory

Example



Angle criterion 90°

Example



Angle criterion 50°

Outline

1. Introduction
2. Trajectory Segmentation
3. Implementation
4. Experimentation
5. Conclusion

Contributions

- A wide state-of-the-art about the trajectory segmentation
- An in-depth analysis of two trajectory segmentation algorithms
- An implementation in MobilityDB of these two algorithms
- An analysis of the performance of the implementation

Future Works

- Improve implemented algorithms:
  - Stable criteria segmentation:
    - Add outlier tolerance
    - Add angle range criterion
  - Sliding window segmentation:
    - Add random walk and cubic spline kernel
- Use external data:
  - Heading, speed from sensors
  - Road and public transportation network
- Add temporal sequence set support

References I

[1] Maike Buchin et al. "Segmenting trajectories: A framework and algorithms using spatiotemporal criteria". In: *Journal of Spatial Information Science* 2011.3 (2011), pp. 33–63.

[2] Sander PA Alewijnse et al. "A framework for trajectory segmentation by stable criteria". In: *Proceedings of the 22nd ACM SIGSPATIAL international conference on advances in geographic information systems*. 2014, pp. 351–360.

[3] Sander PA Alewijnse. "A framework for trajectory segmentation by stable criteria and Brownian bridge movement model". PhD thesis. Master's thesis: Eindhoven University of Technology, 2013.

[4] Boris Aronov et al. "Segmentation of trajectories on nonmonotone criteria". In: *ACM Transactions on Algorithms (TALG)* 12.2 (2015), pp. 1–28.

[5] Mohammad Etemad et al. "SWS: an unsupervised trajectory segmentation algorithm based on change detection with interpolation kernels". In: *GeoInformatica* (2020), pp. 1–21.