AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

FROM SIMPLE FEATURES TO MOVING FEATURES AND BEYOND?

116th OGC Member Meeting, 14-21 September 2020

Anita Graser
AIT Austrian Institute of Technology
University of Salzburg
Vienna / Salzburg, Austria
ORCID: 0000-0001-5361-2885

Esteban Zimányi
Université libre de Bruxelles
Brussels, Belgium
ORCID: 0000-0003-1843-5099

Krishna Chaitanya Bommakanti
Adonmo
HITEC City, Hyderabad,
Telangana 500081, India

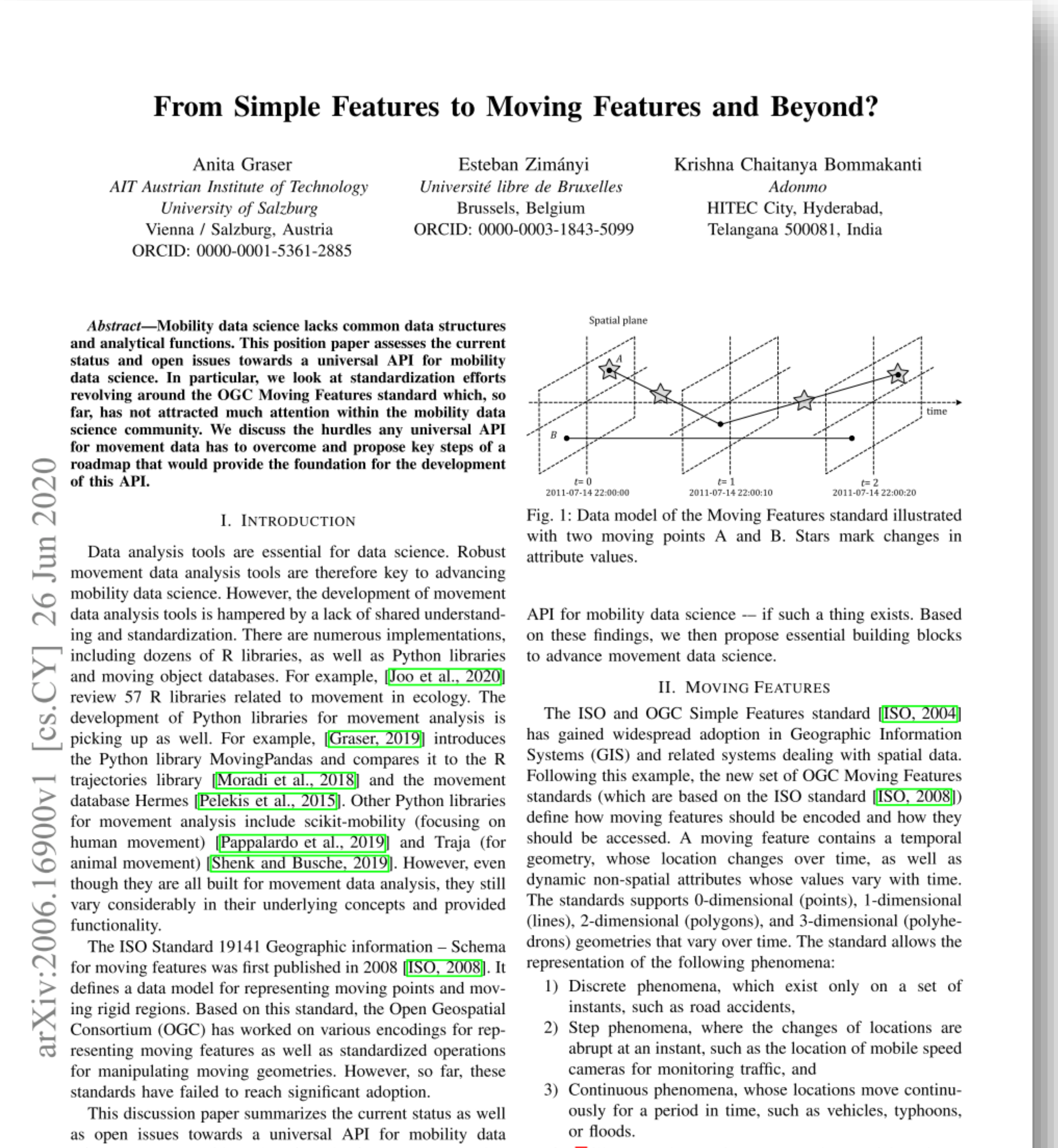
MOTIVATION

Lack of common data structures & analytical functions in **mobility data science**

Assessment of the current status & open issues towards a universal API for mobility data science

Standardization efforts around OGC Moving Features

→ Discussion paper **arXiv:2006.16900**



From Simple Features to Moving Features and Beyond?

Anita Graser
AIT Austrian Institute of Technology
University of Salzburg
Vienna / Salzburg, Austria
ORCID: 0000-0001-5361-2885

Esteban Zimányi
Université libre de Bruxelles
Brussels, Belgium
ORCID: 0000-0003-1843-5099

Krishna Chaitanya Bommakanti
Adonmo
HITEC City, Hyderabad,
Telangana 500081, India

arXiv:2006.16900v1 [cs.CY] 26 Jun 2020

***Abstract*—Mobility data science lacks common data structures and analytical functions. This position paper assesses the current status and open issues towards a universal API for mobility data science. In particular, we look at standardization efforts revolving around the OGC Moving Features standard which, so far, has not attracted much attention within the mobility data science community. We discuss the hurdles any universal API for movement data has to overcome and propose key steps of a roadmap that would provide the foundation for the development of this API.**

I. Introduction

Data analysis tools are essential for data science. Robust movement data analysis tools are therefore key to advancing mobility data science. However, the development of movement data analysis tools is hampered by a lack of shared understanding and standardization. There are numerous implementations, including dozens of R libraries, as well as Python libraries and moving object databases. For example, [Joo et al., 2020] review 57 R libraries related to movement in ecology. The development of Python libraries for movement analysis is picking up as well. For example, [Graser, 2019] introduces the Python library MovingPandas and compares it to the R trajectories library [Moradi et al., 2018] and the movement database Hermes [Pelekis et al., 2015]. Other Python libraries for movement analysis include scikit-mobility (focusing on human movement) [Pappalardo et al., 2019] and Traja (for animal movement) [Shenk and Busche, 2019]. However, even though they are all built for movement data analysis, they still vary considerably in their underlying concepts and provided functionality.

The ISO Standard 19141 Geographic information – Schema for moving features was first published in 2008 [ISO, 2008]. It defines a data model for representing moving points and moving rigid regions. Based on this standard, the Open Geospatial Consortium (OGC) has worked on various encodings for representing moving features as well as standardized operations for manipulating moving geometries. However, so far, these standards have failed to reach significant adoption.

This discussion paper summarizes the current status as well as open issues towards a universal API for mobility data science — if such a thing exists. Based on these findings, we then propose essential building blocks to advance movement data science.

Fig. 1: Data model of the Moving Features standard illustrated with two moving points A and B. Stars mark changes in attribute values.

II. Moving Features

The ISO and OGC Simple Features standard [ISO, 2004] has gained widespread adoption in Geographic Information Systems (GIS) and related systems dealing with spatial data. Following this example, the new set of OGC Moving Features standards (which are based on the ISO standard [ISO, 2008]) define how moving features should be encoded and how they should be accessed. A moving feature contains a temporal geometry, whose location changes over time, as well as dynamic non-spatial attributes whose values vary with time. The standards supports 0-dimensional (points), 1-dimensional (lines), 2-dimensional (polygons), and 3-dimensional (polyhedrons) geometries that vary over time. The standard allows the representation of the following phenomena:

1) Discrete phenomena, which exist only on a set of instants, such as road accidents,
2) Step phenomena, where the changes of locations are abrupt at an instant, such as the location of mobile speed cameras for monitoring traffic, and
3) Continuous phenomena, whose locations move continuously for a period in time, such as vehicles, typhoons, or floods.

BACKGROUND

Anita Graser
anitagraser

movingpandas

movingpandas

Implementation of Trajectory classes and functions built on top of GeoPandas

Python ☆ 335 70

Esteban Zimanyi
estebanzimanyi

MobilityDB

MobilityDB

MobilityDB is a geospatial trajectory data management & analysis platform, built on PostgreSQL and PostGIS.

sql postgresql postgis spatiotemporal

C 15 ☆ 158 1 1 Updated 12 hours ago

Krishna Chaitanya
chaitan94

MEOS

meos

Mobility Engine, Open Source

python cpp geospatial gis mobility spatio-temporal

C++ MIT 0 ☆ 1 15 (12 issues need help) 2 Updated 12 days ago

MOBILITY DATA SCIENCE TOOLS

Robust data analysis tools are essential for advancing mobility data science but **development is hampered by a lack of shared understanding and standardization.**

- Dozens of R / Python libraries + moving object databases
  e.g. 57 R libraries for movement in ecology [Joo et al., 2020]
- Huge variability of underlying concepts & provided functionality

Problems:

→ Waste of resources: each developer group is reinventing the wheel

→ Confusion due to inconsistent terminology

→ Lack of interoperability

STANDARDIZATION

Moving Features supports:

- ✓ 0-dimensional (points)
- ✓ 1-dimensional (lines)
- ✓ 2-dimensional (polygons) and
- ✓ 3-dimensional (polyhedrons) geometries

1. Discrete phenomena
2. Step phenomena
3. Continuous phenomena

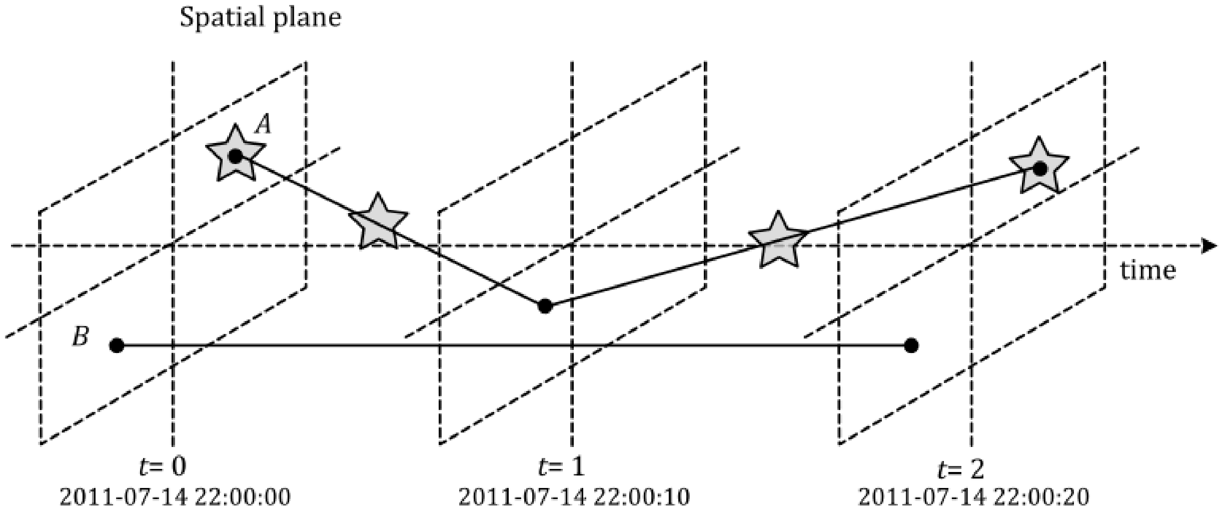
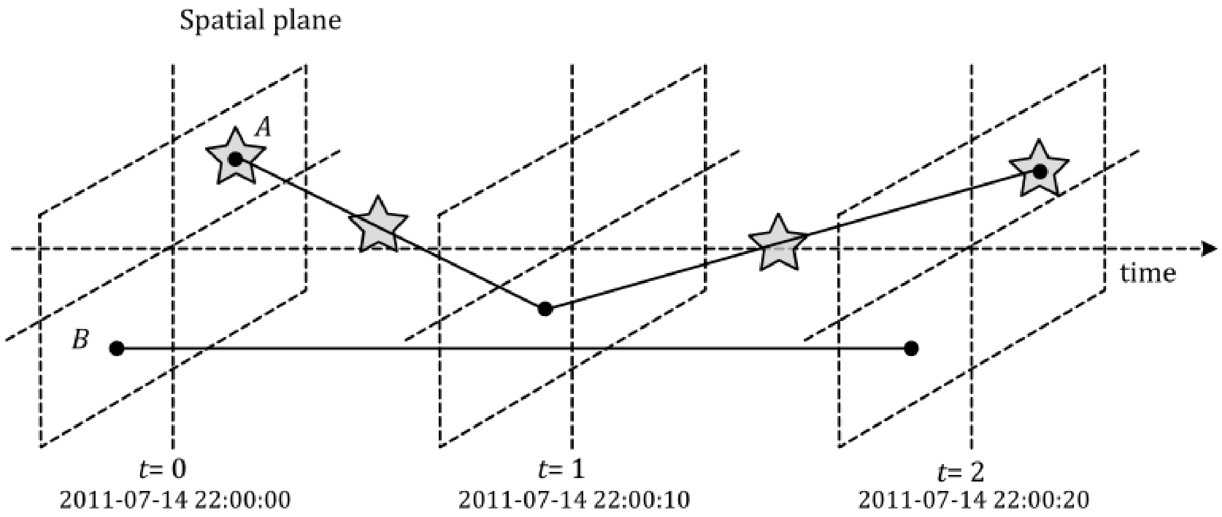


Fig. 1: Data model of the Moving Features standard illustrated with two moving points A and B. Stars mark changes in attribute values.

CSV ENCODING

	CSV
Concept	Segments with start and end time and attributes, no static object properties
Advantages	1) Supports temporal gaps in observations
Limitations	1) Verbose: redundant information (start and end time and location for each segment) 2) Temporal geometry and temporal attributes have to be synced 3) Only linear interpolation 4) Only moving points 5) Not readily usable in GIS (missed the opportunity to use WKT to represent the geometry)




```
@stboundedby,urn:x-ogc:def:crs:EPSG:6.6:4326,
 2D,10.0 10.0,10.6 12.2,2011-07-14T22:00:00Z,
 2011-07-14T22:00:20Z,sec
@columns,mfidref,trajectory,gear,xsd:integer
A,0,5,10.0 10.0 10.2 10.6,1
A,5,10,10.2 10.6 10.4 11.2,2
A,10,15,10.4 11.2 10.5 11.7,2
A,15,20,10.5 11.7 10.6 12.2,3
B,0,20,2.0 2.0 2.1 2.1,1
```

Fig. 2: CSV encoding of vehicles A and B

XML ENCODING

	XML
Concept	Segments with start and end time and attributes, with static properties
Advantages	1) Supports temporal gaps in observations 2) Can handle complex geometries 3) Support for static / non-temporal descriptive object properties
Limitations	1) Very verbose: XML & redundant information (start and end time and location for each segment) 2) Temporal geometry and temporal attributes have to be synced 3) Same interpolation for geometry and all attributes

```
<?xml version="1.0" encoding="UTF-8"?>
 <mf:MovingFeatures xmlns:mf="http://schemas.opengis.net/mf-core/1.0" ...>
 <mf:STBoundedBy offset="sec">
  <gml:EnvelopeWithTimePeriod srsName="urn:x-ogc:def:crs:EPSG:6.6:4326">
   <gml:lowerCorner>10.0 10.0</gml:lowerCorner>
   <gml:upperCorner>10.6, 12.2</gml:upperCorner>
   <gml:beginPosition>2011-07-14T22:00:00Z</gml:beginPosition>
   <gml:endPosition>2011-07-14T22:00:20Z</gml:endPosition>
  </gml:EnvelopeWithTimePeriod>
 </mf:STBoundedBy>
 <mf:Member>
  <mf:MovingFeature gml:id="A">
   <gml:name>NissanA</gml:name>
   <gml:description>Nissan Sentra ...</gml:description>
  </mf:MovingFeature>
 </mf:Member>
 <mf:Header>
  <mf:VaryingAttrDefs>
   <mf:attrDef name="gear" type="xsd:integer">
   <mf:AttrAnnotation>The gear number used... </mf:AttrAnnotation>
   </mf:attrDef>
  </mf:VaryingAttrDefs>
 </mf:Header>
 <mf:Foliation>
  <mf:LinearTrajectory gml:id="LT0001" mfIdRef="A" start="0" end="5">
   <gml:posList>10.0 10.0 10.2 10.6</gml:posList>
   <mf:Attr>1</mf:Attr>
  </mf:LinearTrajectory>
  <mf:LinearTrajectory gml:id="LT0003" mfIdRef="A" start="5" end="10">
   <gml:posList>10.2 10.6 10.4 11.2</gml:posList>
   <mf:Attr>2</mf:Attr>
  </mf:LinearTrajectory>
  <mf:LinearTrajectory gml:id="LT0003" mfIdRef="A" start="10" end="15">
   <gml:posList>10.4 11.2 10.5 11.7</gml:posList>
   <mf:Attr>2</mf:Attr>
  </mf:LinearTrajectory>
  <mf:LinearTrajectory gml:id="LT0003" mfIdRef="A" start="15" end="20">
   <gml:posList>10.5 11.7 10.6 12.2</gml:posList>
   <mf:Attr>3</mf:Attr>
  </mf:LinearTrajectory>
 </mf:Foliation>
</mf:MovingFeatures>
```

Fig. 3: XML encoding of vehicle A

JSON ENCODING

	JSON
Concept	Points with timestamps and attributes with (independent) timestamps
Advantages	1) Most compact representation 2) Can handle complex geometries 3) Time stamps of location and attribute changes are modelled independently – no synchronization necessary 4) Interpolation modes can be specified individually for each attribute 5) Support for non-temporal descriptive attributes
Limitations	1) Multiple options for encoding the same situation (e.g. unclear bounds of time periods) 2) No support for temporal gaps in observations


```
{
  "type": "MovingFeature",
  "temporalGeometry": {
     "type": "MovingPoint",
     "coordinates": [ [10.0, 10.0], [10.4, 11.2], [10.6, 12.2] ],
     "datetimes": ["2011-07-14T22:00:00Z", "2011-07-14T22:00:10Z", "2011-07-14T22:00:20Z"],
     "interpolations": "Linear"
  },
  "temporalProperties": [ {
     "name": "gear",
     "values": [1, 2, 3, 3],
     "datetimes": ["2011-07-14T22:00:00Z", "2011-07-14T22:00:05Z", "2011-07-14T22:00:15Z",
                   "2011-07-14T22:00:20Z"],
     "interpolations": "Stepwise"
  }, ],
  "stBoundedBy": {
     "bbox": [10.0, 10.0, 10.6, 12.2],
     "period": { "begin": "2011-07-14T22:00:00Z", "end" : "2011-07-15T22:00:20Z" }
  },
  "properties": {
     "name": "NissanA", "description": "Nissan Sentra ..."
  }
}
```


Fig. 4: JSON encoding of vehicle A

ENCODING SUMMARY

	CSV	XML	JSON
Concept	Segments with start and end time and attributes, no static object properties	Segments with start and end time and attributes, with static properties	Points with timestamps and attributes with (independent) timestamps
Advantages	1) Supports temporal gaps in observations	1) Supports temporal gaps in observations 2) Can handle complex geometries 3) Support for static / non-temporal descriptive object properties	1) Most compact representation 2) Can handle complex geometries 3) Time stamps of location and attribute changes are modelled independently – no synchronization necessary 4) Interpolation modes can be specified individually for each attribute 5) Support for non-temporal descriptive attributes
Limitations	1) Verbose: redundant information (start and end time and location for each segment) 2) Temporal geometry and temporal attributes have to be synced 3) Only linear interpolation 4) Only moving points 5) Not readily usable in GIS (missed the opportunity to use WKT to represent the geometry)	1) Very verbose: XML & redundant information (start and end time and location for each segment) 2) Temporal geometry and temporal attributes have to be synced 3) Same interpolation for geometry and all attributes	1) Multiple options for encoding the same situation (e.g. unclear bounds of time periods) 2) No support for temporal gaps in observations

TABLE I: Overview of OGC Moving Feature encodings

ACCESS

Functions for

- ✓ Retrieval of trajectory feature attributes, e.g.
  - ✓ Location, speed, or acceleration at a given time
  - ✓ Subtrajectory between two timestamps
- ✓ Operations between trajectories and geometry objects, e.g.
  - ✓ Time at a given point

ACCESS

Functions for

- ✓ Retrieval of trajectory feature attributes, e.g.
  - ✓ Location, speed, or acceleration at a given time
  - ✓ Subtrajectory between two timestamps
- ✓ Operations between trajectories and geometry objects, e.g.
  - ✓ Time at a given point

Open questions, e.g.

- **TimeToDistance** *“shall return a graph of the time to distance function as a set of curves in the Euclidean space consisting of coordinate pairs of time and distance”*

IDENTIFIED CHALLENGES

Part 1: Moving Features standard

1. Limited public awareness of the standard's existence
2. Data model limitations (particularly of the CSV and XML encodings)
3. Multiple encoding standards with different concepts
4. Ambiguous situations / no official reference implementation

IDENTIFIED CHALLENGES

Part 1: Moving Features standard

1. Limited public awareness of the standard's existence
2. Data model limitations (particularly of the CSV and XML encodings)
3. Multiple encoding standards with different concepts
4. Ambiguous situations / no official reference implementation
5. Lack of WKT / WKB extensions for straight-forward integration into existing systems

Part 2: API for Mobility Data Science

1. The curse of movement data heterogeneity
2. Lack of pragmatic solutions
3. Wrong priorities and incentives

VISION

Essential building blocks for an API for Mobility Data Science

1. Common movement data analysis concepts
2. An open general-purpose mobility (data) engine
3. Focus on open science and reproducibility

AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

ANITA GRASER

anita.graser@ait.ac.at

@underdarkGIS