Exploratory Trajectory Analysis for Massive Historical AIS Datasets

Anita Graser*†, Melitta Dragaschnig*, Peter Widhalm*, Hannes Koller* and Norbert Brändle*
*AIT Austrian Institute of Technology, Vienna, Austria
†University of Salzburg, Salzburg, Austria
Email: anita.graser@ait.ac.at
ORCID: AG: 0000-0001-5361-2885, MD: 0000-0001-5100-2717, PW: 0000-0002-5074-5356, HK: 0000-0002-4255-3530, NB: 0000-0002-2976-3138

***Abstract*—Data exploration is an essential task for gaining an understanding of the potential and limitations of novel datasets. This paper discusses the challenges related to exploring large Automatic Identification System (AIS) datasets. We address these challenges using trajectory-based analysis approaches implemented in distributed computing environments using Spark and GeoMesa. This approach enables the exploration of datasets that are too big to handle within conventional spatial database systems. We demonstrate our approach using a case study of 4 billion AIS records.**

***Index Terms*—exploratory data analysis, mobility data, movement data, travel time, spatiotemporal**

I. Introduction

Massive ship movement datasets collected from the Automatic Identification System (AIS) have the potential to improve maritime safety and efficiency of operations. Big AIS datasets can serve as input for machine learning approaches, for example, to extract ship routes and predict travel times [1] or future destinations [2]. The performance and therefore success of data-driven approaches depends strongly on the quality of the input data, that is the suitability of the data for a certain purpose. "Garbage in – garbage out" is a well known concept in computer science and mathematics. Therefore, it is necessary to evaluate data suitability.

Exploratory data analysis (EDA) [3] analyzes data sets to determine what information the data contains. This is commonly achieved by summarizing the dataset's main characteristics, often using visualizations. EDA goals are to assess assumptions and suggest hypotheses, to select statistical tools, and to provide a basis for further data collection if required. EDA concepts for movement data have been covered extensively by [4].

A. Problem statement

The majority of existing movement data analysis methods cannot deal with large datasets since, in the past, movement data analysis had to deal with limited data availability. Therefore, traditional approaches quickly reach their limits as dataset size increases. Since the limits of existing tools for storing, processing, and visualizing movement data vary, there is no one clear definition of the term "massive" in the context of movement data analysis. However, the common denominator is that big or massive datasets cannot be handled by conventional tools on a single machine.

To get a feeling for where the limits of conventional tools may lie, we refer to the literature. For example, [5] report a processing time of one day to create vessel tracks and density maps of 60 thousand AIS records (one month of AIS information from around Shetland) using ArcGIS. For a global ship density grid using 1.5 billion AIS records, [6] report a processing time of "about a week" using PostGIS and GDAL. These lengthy processing times limit the extent of data exploration that analysts can perform within a given time frame.

Sampling is a common approach to reduce datasets to a size that can still be handled. However, it is hard to extract useful data samples for movement exploration tasks. To explore a dataset and assess preliminary assumptions about the data, it is necessary to be able to look at the whole dataset since sampling can reinforce assumptions.

While the above mentioned challenges concern all movement datasets, maritime movement data presents additional challenges. Maritime vessel trip properties vary significantly, for example, regarding duration (from minutes to weeks) and spatial extent (from local to global). Furthermore, AIS data sources usually do not provide complete global coverage without gaps but are limited to certain regions. Therefore, vessel AIS tracks may contain long observation gaps. Additionally, reported information on movement speed and direction, trip destination [5], vessel identity [6], time stamps [7] and more are not always reliable.

To address these challenges, it is necessary to develop distributed computing approaches for maritime movement data. Developments in this direction include, for example [8]–[11].

B. Contribution

There is a lack of established EDA tools as well as a lack of literature on best practices for applying EDA to movement data in general [12] and AIS data in particular. To address the current lack of best practices, this paper proposes concepts for the systematic exploration of large AIS datasets. We demonstrate these concepts using a case study of a dataset with 4 billion records. We cover analyses ranging from raw AIS records (II-A), to continuous vessel tracks (II-B) and, finally

This work was supported by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT) within the programme "IKT der Zukunft" under Grant 861258 (project MARNG).

trajectories of individual vessel journeys with meaningful start and end locations at harbors or anchoring sites (II-C). To demonstrate exploration for a specific machine learning task, we try to assess the potential for travel time prediction based on trajectories (II-D).

The remainder of this paper is structured as follows: Sec. II presents our proposed AIS data exploration concepts including establishing a first overview of the data, considering continuous movement tracks, and finally extracting semantically meaningful trajectories and their potential for travel time modelling. Finally, Sec. III sums up our conclusions and provides an outlook for future work.

II. Exploring 4 billion AIS records

In our case study, we use AIS data published by the Danish Maritime Authority[1] for the year 2017. Each one-day CSV file has a size of around 2 GB (uncompressed). In total, the dataset consists of 4 billion records with 22 attributes.

To store this data for distributed processing, we use GeoMesa Accumulo. GeoMesa provides fast spatiotemporal indexing [14] to help store and access spatiotemporal data. GeoMesa also provides spatial analysis functions that can be called by Apache Spark. Spark is a well-established general-purpose cluster-computing framework first proposed by [13]. GeoMesa was chosen due to its mature support for spatial vector data and the ability to publish data stored in GeoMesa via GeoServer using standardized OGC web services, such as WMS and WFS. By supporting these standards, any standards-compliant client application can be used to visualize the data.

A. Establishing an Overview

The first step in our data exploration framework is to establish an overview by exploring the raw input data records and comparing the results with our assumptions. This includes summary statistics of different record attributes, as well as an assessment of spatiotemporal extent and gaps in the data.

Looking at the data attributes, for example, we find that the dataset contains records from 89,926 distinct MMSIs (Maritime Mobile Service Identity, a nine-digit id which identifies ships and other maritime objects; MMSIs should be unique but user error can lead to multiple vessels using the same identifier).

Concerning the *temporal extent* of the dataset and potential gaps, we assume that the dataset should cover the whole year without gaps. Indeed, results show that the number of records per day is quite stable, usually ranging between 10 and 12 million records per day. There are no longer periods without any data.

Concerning the *spatial extent*, we assume that our dataset should cover Danish waters and surrounding areas. However, results show many points outside of the expected region. This could be because of data errors since some minimum and maximum latitude and longitude values seem implausible. It could also mean that the dataset covers a larger than expected area. To check if individual records are part of plausible movements, it is necessary to aggregate records into continuous tracks.

[1]Data source: ftp://ftp.ais.dk/ais_data/

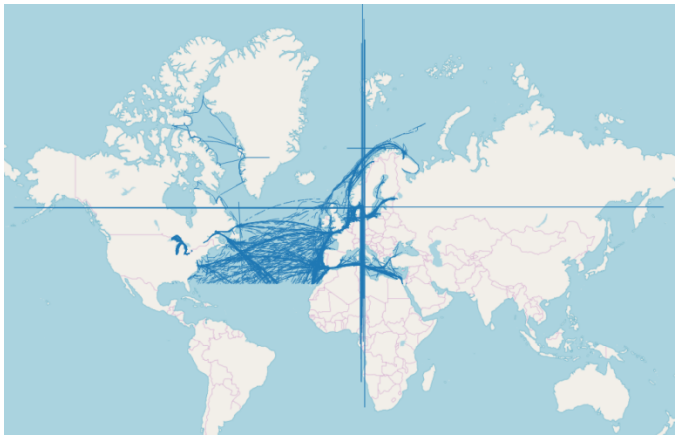

Fig. 1. Spatial extent: raw one-day tracks of 265 vessels on long-distance voyages grouped by MMSI and date. (Background map ©OpenStreetMap contributors.

B. Aggregating Continuous Tracks

The next step is to connect the location records of individual vessels to form continuous tracks. A common approach used when working with spatial databases is to aggregate AIS records into LineStrings by grouping the chronologically sorted records by MMSI (and optionally by date or another time unit). This approach can be translated to Spark (or SparkSQL). Based on the assumption that many vessels operate only in a small area, but some should travel long distances, including voyages between different continents, we selected all records of vessels that reported at least one destination in Asia in 2017. (Since destination is a free text field, a list of potential destination names was compiled manually.) Fig. 1 shows resulting track lines for the 265 vessels in this sample. The pronounced vertical and horizontal lines in Fig. 1 indicate errors in the position data that require data cleaning. Furthermore, tracks crossing the North Atlantic and Mediterranean show that the dataset covers a much larger area than expected. (Further inspection shows that positions in the Atlantic are not available for the whole year but are limited to the period between May and December 2017.)

Tracks generated by grouping AIS records by MMSI and date aggregate all movements of a vessel during one day. However, these tracks do not represent semantically meaningful trajectories (individual vessel journeys) with a start and end at a meaningful location (harbor or other anchoring site). Meaningful trajectories can be extracted by splitting continuously observed movement tracks at stops, which have to be detected. Additionally, since AIS sources are limited to a certain coverage area, they cannot ensure continuous observations. Therefore, tracks should also be split at observation gaps to avoid misinterpretations, such as, for example, exaggerated travel times due to an unobserved stop.

Fig. 2. Full set of trajectories between July and December 2017 (n=144, 870).

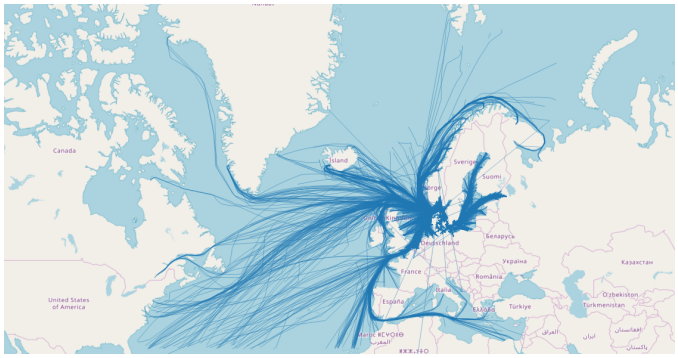

Fig. 3. Trajectories passing Gothenburg between July and December 2017.

Since a single vessel journey can take weeks, it would be insufficient to generate daily tracks and to split them into individual trajectories afterwards. Instead, longer time spans would have to be aggregated (resulting in longer computation times and potential memory issues) or, alternatively, trajectories would have to be stitched together in a later processing step. To address this issue, we propose an iterative approach that can build trajectories quickly and does not run out of memory (as long as individual trajectories fit into memory).

C. Extracting Trajectories

Our proposed trajectory aggregation approach solves some interesting challenges: 1) processing massive amounts of trajectory data where 2) operations only produce correct results if applied to a complete and chronologically sorted group of position records. Our implementation is based on the Secondary Sort pattern [15] and on Spark's aggregator concept. With Secondary Sort, records are first grouped by a key, and when iterating over all records within a group, the records will be sorted by a certain criterion. The result of this process, an iterator, is then used by an aggregator that implements the logic required to build trajectories based on gaps and stops detected in the dataset.

It has to be noted that building trajectories using Spark core libraries[2] can be challenging: Spark provides high-level functions for grouping and aggregating records, however these are mostly geared towards dealing with unsorted data. When using high-level Spark core functionality incorrectly, an aggregator needs to collect and sort the entire trajectory in the main memory of a single processing node. Consequently, when dealing with large datasets, out-of-memory errors are frequently encountered. The third-party library spark-sorted[3] provides a higher-level abstraction by wrapping the complex low-level Spark functionality, thus allowing us to: 1) group a massive dataset by a key 2) sort the records within each group by timestamp and 3) iteratively process the sorted records of the group in such a way that they never need to be materialized in memory all at once. As shown in the Scala code skeleton provided in the appendix, iterator-based streaming of the sorted data is provided by spark-sorted's groupSort function (line 11). A trajectory can be aggregated (line 12) one position at a time (line 35). Resampling, compression, and segmentation can be applied on-the-fly as needed: Whenever a new position is processed (line 51), the trajectory can be checked for a segmentation criterion, such as an observation gap or detected stop (line 52). If a segmentation criterion is detected, the current trajectory can be finalized and a new one can be started at the current position (line 40). This succinct approach supports processing movement datasets of arbitrary size within the Spark framework.

Fig. 2 shows the resulting trajectories for the time between July and December 2017 (which corresponds to the time range for which position data in the North Atlantic is available). The minimum trajectory duration is set to 1 hour. Shorter trajectories are discarded for this exploratory analysis. Trajectories are split at observation gaps that are longer than 24 hours. Furthermore, trajectories are down-sampled to 1 hour sampling intervals. This down-sampling mostly affects sections of trajectories that fall within the terrestrial AIS coverage since satellite AIS reporting intervals are already long. Additionally, outliers resulting in unrealistic speeds were removed. This process results in $144,870$ trajectories and takes around 1 hour to compute[4]. Of course, the number of trajectories varies based on the minimum duration and split criteria.

For storage and visualization purposes, trajectories are commonly summarized using MMSI, start time, end time, and LineString geometry. The resulting trajectories provide a convenient way to access continuously observed vessel voyages. For example, Fig. 3 shows trajectories filtered based on a spatial intersection with a bounding box surrounding Gothenburg, Sweden. (Gothenburg was chosen as one of the focus areas of our exploration efforts since it is one of the busiest ports in the region around Denmark.) However, using this trajectory model, it is only possible to determine travel times between trip start and end locations but not for arbitrary

[2]Spark 2.4.5, at the time of writing

[3]https://github.com/tresata/spark-sorted

[4]Our cluster comprises six data nodes: three nodes with two Intel Xeon E5-2430L CPUs and 32G RAM, and three nodes with two Intel Xeon E5-2660 v3 and 64G RAM. Operating system and HDFS file system reside on SSDs. The setup is based on Apache Hadoop 2.7 and managed with Ambari 2.6.

Fig. 4. Trajectories between Gothenburg and Gibraltar between July and December 2017. Red dots mark trajectory start and end locations.

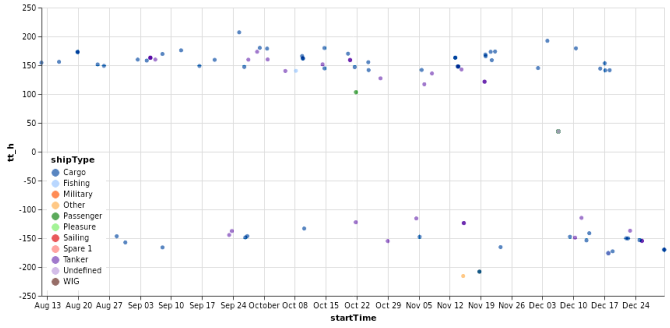

Fig. 5. Travel times between Gothenburg and Gibraltar [in hours] (positive: travel time Gothenburg to Gibraltar, negative: vice versa).

TABLE I
TRAVEL TIME STATISTICS FOR THE ROUTE BETWEEN GOTHENBURG AND GIBRALTAR [IN HOURS].

	to Gibraltar	to Gothenburg
Min	35	115
Q1	144	141
Median	157	149
Q3	168	157
Max	207	216
Mean	154	152
Stdev	25	23

locations along the trajectory. This means that a lot of travel time information is lost. To avoid this loss of information, we propose to also store time information along the trajectory. In spatial databases like PostGIS, this can be achieved by storing time stamps in the measure value of LineStringM features [16]. GeoMesa, however, currently does not provide support for LineStringM geometries. We therefore opt for storing the sequence of time stamps in a separate field.

D. Assessing the Potential for Travel Time Prediction

Data-based travel time predictions can improve port operations such as pilot and berth planning, as well as other hinterland logistics operations. Research is targeting both long-term and short-term predictions which aim to predict the arrival date or the more specific arrival time, respectively.

a) Long-term predictions: As previous analysis steps showed, our dataset has the potential to help train prediction models for voyages crossing the North Atlantic and the Mediterranean. We can apply both spatial and temporal filters to the extracted trajectories to determine how much training data is available for a certain origin/destination combination.

For example, on the route between Gothenburg and Gibraltar (Fig. 4), we detected 83 trips, mostly by cargo vessels and tankers. (Gibraltar was chosen due to its strategically important location at the entrance to the Mediterranean.) The corresponding travel times are shown in Fig. 5. The visualization shows no clear seasonal trends in the travel times. Mean travel times are 154±25 or 152±23 hours (around 6 days), depending on the direction (Tab. I). The map view (Fig. 4) shows one clear outlier route visible as a straight line crossing France, as well as one vessel circumnavigating the UK in the North.

b) Short-term predictions: For the analysis of short-distance trips, the previously used aggregation with a resampling interval of 1 hour is too coarse. Therefore, a second set of trajectories was computed with a minimum gap duration of 15 minutes and a resampling interval of 1 minute. Additionally, the analysis area was restricted to the Skagerrak/Kattegat region (including most of the waters between Denmark and Sweden). Even though the resampling interval is only 1 minute, this still reduces the dataset size considerably, since all records of stationary vessels are removed by the stop detection process and segments with dense reporting intervals are thinned out.

Looking at the resulting trajectories, for example, the route between Gothenburg and Kiel is mostly travelled by passenger vessels. Mean travel times are 14±0.6 hours. Travel times do not exhibit seasonal variations but they do vary slightly by day of the week, as shown in Fig. 6. Indeed, an investigation into official ferry schedules reveals that the ferry trip should take 14.5h on weekdays and 15.5h on weekends, thus confirming our data exploration results. Longer travel times and larger variations are observed for other ship types. A travel time prediction model for passenger vessels on this route would therefore provide better results than for other vessel types.

Beyond vessel type, AIS also provides other vessel information such as its size (length and width) or typical speeds (SOG). We assume that size and typical speed should be valuable explanatory variables for travel time prediction models. To test this assumption, we compute typical speeds for each vessel, by determining the approximate 90^{th} percentile of reported SOG where $SOG > 1$. Fig. 7 shows the correlation between typical vessel speeds (shipSog) and travel times for the route between Gothenburg and Helsingborg.

By fitting a linear regression model to explain travel time using the 90^{th} speed percentiles (SOG90%), we can determine that up to 33% of the total variation on this route is explained by the 90^{th} speed percentile. The relative RMSE (rRMSE)

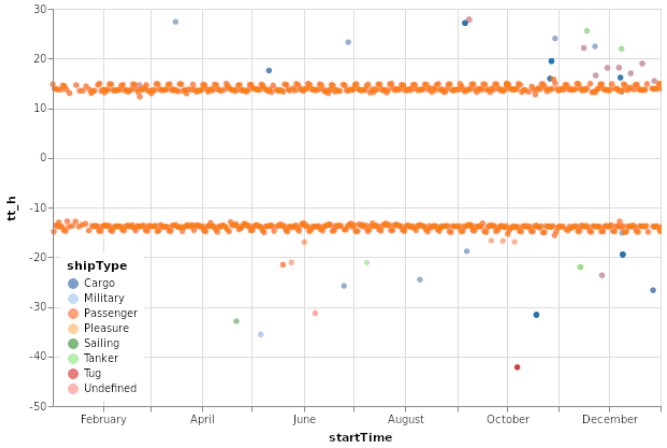

Fig. 6. Travel times between Gothenburg and Kiel [in hours].

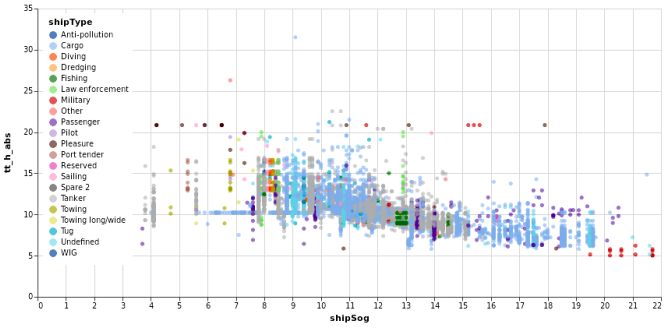

Fig. 7. Travel times between Gothenburg and Helsingborg over vessel speed [in hours over knots].

ranges between 17.2% and 19.5% (Tab. II). Adding ship length (discarding records with $length > 200m$) to the model further improves the results to rRMSEs between 15.8% and 17.6% (Tab. III).

While these results indicate that it may be promising to build a predictor using these variables, care should be taken not to over-generalise based on individual routes. However, these exploratory analyses provide valuable input for feature engineering and provide grounds for discussion with domain experts.

III. Conclusion and Outlook

We have presented an exploratory data analysis for massive AIS datasets, including concrete examples of exploring a dataset of 4 billion records using distributed computing approaches. We have proposed core concepts for a three step EDA process that covers: 1) establishing an overview by exploring raw movement data records, 2) exploring large-scale movement patterns by aggregating continuous movement tracks, and 3) understanding connections by extracting trajectories of individual vessel journeys between meaningful start and end locations.

Since there is a general lack of established EDA tools for movement data, the technical implementation of the exploration is up to the individual analyst or researcher. In this context, we have discussed various challenges that need to be overcome to perform AIS analyses, including large variations in trajectory length and duration, observation gaps, as well as data errors. We have proposed trajectory aggregation approaches that account for these issues. Finally, we demonstrated how EDA can support feature engineering for machine learning, using maritime travel time prediction as an example.

Recent developments regarding trajectory data handling in distributed environments (for example [17]) underline the continued need for movement data specific tools that can deal with large amounts of data. To support analysts and researchers with assessing data suitability for different purposes, these data indexing and querying tools need to be combined with solid exploration concepts. However, the heterogeneity of movement data applications and datasets presents a major challenge for the development of general-purpose data exploration tools [12]. Therefore, domain-specific developments will be necessary to provide exploration solutions that can be efficiently used in the maritime domain.

TABLE II
Linear regression model SOG90th training summary

	All types	Cargo	Tanker
Coefficient	-0.397	-0.3878	-0.332
Intercept	15.274	15.033	14.467
RMSE	2.059	1.998	1.857
rRMSE	19.3%	19.5%	17.2%
r2	0.299	0.328	0.197

TABLE III
Linear regression model SOG90th+length training summary

	All types	Cargo	Tanker
Coefficient	-0.231/-0.026	-0.166/-0.0372	-0.121/-0.030
Intercept	15.949	16.435	14.989
RMSE	1.878	1.704	1.700
rRMSE	17.6%	16.6%	15.8%
r2	0.417	0.515	0.315

References

[1] I. Parolas, L. Tavasszy, and I. Kourounioti, "Prediction of Vessel's Estimated Time of Arrival (ETA) in Container Terminals – A Case Study in the Port of Rotterdam." in TRB 96th Annual Meeting, 2017.
[2] G. Pallotta, M. Vespe, and K. Bryan, "Vessel pattern knowledge discovery from AIS data: A framework for anomaly detection and route prediction," Entropy, vol. 15(6), pp. 2218-–2245, 2013.
[3] J. W. Tukey, Exploratory Data Analysis. Addison-Wesley, 1977.
[4] G. Andrienko, N. Andrienko, P. Bak, D. Keim, and S. Wrobel, Visual analytics of movement. Springer, Berlin, 2013.
[5] R. L. Shelmerdine "Teasing out the detail: How our understanding of marine AIS data can better inform industries, developments, and planning," Marine Policy, vol. 54, pp.17-–25, 2015.
[6] M. Aronsen and K. Landmark, Density mapping of ship traffic. FFI-RAPPORT 16/02061, 2016.
[7] K. Patroumpas, E. Alevizos, A. Artikis, M. Vodas, N. Pelekis, and Y. Theodoridis, "Online event recognition from moving vessel trajectories," GeoInformatica, vol. 21(2), pp. 389–427, 2017.
[8] G. Spiliopoulos, D. Zissis, and K. Chatzikokolakis, "A big data driven approach to extracting global trade patterns." in Mobility Analytics for Spatio-Temporal and Social Data. MATES 2017. LNCS, vol. 10731, pp. 109–121. Springer, Cham, 2017.

[9] D. Zissis, K. Chatzikokolakis, G. Spiliopoulos, and M. Vodas, "A Distributed Spatial Method for Modeling Maritime Routes," in IEEE Access, vol. 8, pp. 47556–47568, 2020.
[10] L.M. Millefiori, D. Zissis, L. Cazzanti, and G. Arcieri, "A distributed approach to estimating sea port operational regions from lots of AIS data." in IEEE Big Data, pp. 1627–1632, 2016.
[11] A. Graser, P. Widhalm, and M. Dragaschnig, "The M³ massive movement model: a distributed incrementally updatable solution for big movement data exploration." Int. J. Geogr. Inf. Sci., in press.
[12] A. Graser and M. Dragaschnig, "Open Geospatial Tools for Movement Data Exploration," KN J. Cartogr. Geogr. Inf., 2020.
[13] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica "Spark: cluster computing with working sets." in HotCloud, p.10, 2010.
[14] J. N. Hughes, C. N. Eichelberger, A. Fox, A. Hulbert, and M. Ronquest, "GeoMesa: a distributed architecture for spatio-temporal fusion," in Geospatial Informatics, Fusion, and Motion Video Analytics V, vol. 9473, p. 94730f, Int. Soc. for Optics and Photonics, 2015.
[15] M. Parsian, Data Algorithms. O'Reilly Media, 2015.
[16] A. Graser, "Evaluating Spatio-temporal Data Models for Trajectories in PostGIS Databases," GI_Forum Journal, vol. 1-2018, pp. 16–33, 2018.
[17] R. Li, H. He, R. Wang, Y. Huang, J. Liu, S. Ruan, ... and Y. Zheng, "JUST: JD Urban Spatio-Temporal Data Engine," ICDE. IEEE, 2020.

IV. Appendix

```scala
import com.tresata.spark.sorted.PairRDDFunctions._

object CreateTrajectories {
 def main(args: Array[String]): Unit = {
  val spark: SparkSession = SparkSession.builder.appName("CreateTrajectories").getOrCreate()
  import spark.implicits._

  val aisRecordsRDD: RDD[(Int, AISRecord)] = Source.getRecords()  // RDD of (MMSI, AISRecord)

  val result: Dataset[Trajectory] = aisRecordsRDD
   .groupSort(Ordering.by[AISRecord, Long](_.timestamp.getTime)) // group by mmsi, sort by time
   .mapStreamByKey {toTrajectoryIterator(_)}.values.toDS()
  }

 def toTrajectoryIterator(itr: Iterator[AISRecord]): TrajectoryAggregator = {
  val trajectorybuilder: TrajectoryBuilder = TrajBuilderFactory.newTrajectoryBuilder(conf)
  new TrajectoryAggregator(itr, trajectorybuilder)
 }
}

class TrajectoryAggregator(sortedAISrecords: Iterator[AISRecord],
                           trajectoryBuilder: TrajectoryBuilder) extends Iterator[Trajectory] {
 var nextTrajectory: Trajectory = createNext()

 override def hasNext(): Boolean = nextTrajectory != null

 override def next(): Trajectory = {
  val currentResult: Trajectory = nextTrajectory
  nextTrajectory = createNext()
  currentResult
 }

 def createNext(): Trajectory =  {  // create and return the next trajectory
  while (sortedAISrecords.hasNext) {
   val record: AISRecord = sortedAISrecords.next()

   if (trajectoryBuilder.shouldTrajectoryInclude(record)) {
    trajectoryBuilder.addRecord(record)
   } else {  // finish the current trajectory and start a new one
    val result:Trajectory = trajectoryBuilder.buildAndReset()
    trajectoryBuilder.addRecord(record)
    return result
   }
  }
  // all records have been processed. Create a final trajectory, if any
  trajectoryBuilder.buildAndReset()
 }
}

trait TrajectoryBuilder {
 def addRecord(r: AISRecord) // performs resampling
 def shouldTrajectoryInclude(r: AISRecord): Boolean  // handles gap and stop detection
 def buildAndReset(): Trajectory
}
```