Extracting Patterns from Large Movement Datasets

GI_Forum 2020, Issue 1
Page: 153 - 163
Short Paper
Corresponding Author:
anita.graser@ait.ac.at
DOI: 10.1553/giscience2020_01_s153

Anita Graser[1,2], Peter Widhalm[1] and Melitta Dragaschnig[1]

[1]AIT Austrian Institute of Technology, Vienna, Austria
[2]University of Salzburg, Salzburg, Austria

Abstract

Extracting useful information from large spatiotemporal datasets is a challenging task that requires suitable visual data representations. Big movement data are particularly hard to visualize since they are prone to visual clutter caused by overlapping and crisscrossing trajectories. Different data aggregation approaches have been developed to address this challenge and to provide analysts with better visualizations for data exploration and data-driven hypothesis generation. However, most approaches for extracting patterns, such as mobility graphs or generalized flow maps, cannot handle large input datasets. This paper presents a flow extraction algorithm that can be used in distributed computing environments and thus make it possible to explore movement patterns in large datasets. We demonstrate its usefulness in a use case exploring maritime vessel movements

Keywords:

trajectories, spatiotemporal analysis, movement data analysis

1 Introduction

Large movement datasets that are collected by systems tracking vehicles, people or goods have the potential to improve our understanding of mobility and transport systems, in order to, for example, monitor vehicle emissions or tackle the issue of rising road traffic fatalities (WHO, 2018). Data exploration and data-driven hypothesis generation are important steps in the process of building data-driven models since they enable knowledge to be gained, and spatial modelling (Miller & Goodchild, 2015). However, we humans are not well equipped to understand large amounts of raw numerical data. Instead, we need to visually represent the data to extract useful information. The development of visualizations of big spatial data, however, is challenging (Robinson et al., 2017). Movement data in particular are hard to visualize due to visual clutter caused by intersecting and overlapping trajectories. Therefore, it is 'necessary to use appropriate data abstraction methods' (Andrienko & Andrienko, 2011).

Data aggregation is a common technique for dealing with large amounts of data (Andrienko et al., 2017a). Concerning movement data, density surfaces are probably the most commonly used aggregation technique, as can be inferred from their prevalence in review (Chen et al., 2015; Andrienko et al., 2017b; He et al., 2019) and application papers (Willems et al., 2009; Aronsen & Landmark, 2016). The temporal dimension has been integrated into density concepts in Demšar & Virrantaus' (2010) space-time density volumes of trajectories. However, density approaches provide only limited data exploration capabilities.

More advanced aggregation techniques aim to extract mobility graphs or generalized flow maps from movement records. For example, Andrienko & Andrienko (2011) extract and cluster characteristic waypoints from trajectories to generate aggregated flow maps, as illustrated in Figure 1. However, their approach cannot deal with large datasets. As a work-around, they therefore suggest extracting and clustering characteristic points from a subset of the full trajectory dataset. 'Assuming that the sampling is done sufficiently well, i.e., the statistical and spatial distribution properties of the whole data set are preserved in a sample, we can use the territory division so obtained to summarize […] the whole database' (p. 216).

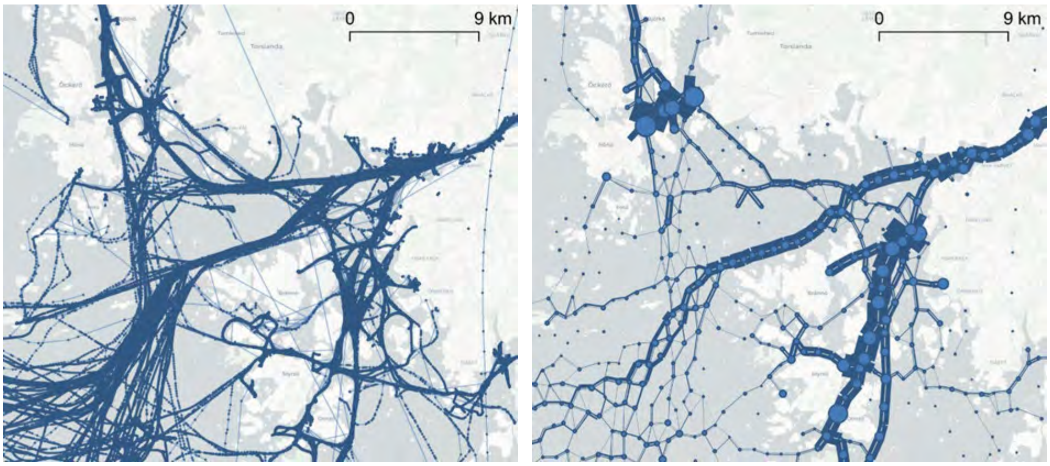

Figure 1: Example of raw movement data (left) and extracted flow map (right) using the algorithm by Andrienko & Andrienko (2011), applied to one day of vessel movement data in the area surrounding Gothenburg, Sweden. The flow map clearly communicates the relative popularity of the different route options in this area. (Background map: Positron © OpenStreetMap contributors and CARTO)

Extracting suitable samples from large movement datasets is no simple task. To avoid sampling, Pallotta et al. (2013) instead use incremental DBSCAN to identify waypoints. However, tuning DBSCAN parameters 'for good waypoint identification is not possible when dealing with areas with varying density' (Dobrkovic et al., 2018, p. 25). Approaches addressing the issue of varying density include lattice or grid-based DBSCAN (Xiao et al., 2017) as well as other clustering algorithms, such as OPTICS (Rinzivillo et al., 2008) or genetic algorithms (Dobrkovic et al., 2018). In Graser et al. (2020), we present M^3 – a movement data exploration model that uses an incremental grid-based clustering algorithm. M^3 runs in distributed computing environments and is therefore scaleable to large datasets that exceed the processing capacity of individual machines. Besides location information, M^3 also takes other movement characteristics, such as speed and direction, into account. However, Graser et al. (2020) do not cover the follow-up step of computing flows. To address this gap, this paper proposes an algorithm for computing flows from massive movement datasets.

The remainder of this paper is structured as follows: Section 2 presents our incremental flow computation algorithm; Section 3 presents a case study with massive vessel movement data; finally, Section 4 draws conclusions and provides an outlook for future work.

2 Methodology

Conceptually similar to Andrienko & Andrienko (2011), our proposed flow extraction method is based on a two-step process. First, we extract prototypes from the movement data. These prototypes describe movement characteristics in a certain geographic area and contain the following information:

- Number of input location records (similar to density surfaces but with support for multiple prototypes per grid cell)
- Geographical distribution (mean coordinates and variance) of location records
- Distributions (mean and variance) of direction, speed and other characteristics available in the location records (including temporal or seasonal information).

In the second step, we determine flows between prototypes, including information about:

- Distribution of travel speeds
- Number of observed transitions.

The details of both steps are described in the following subsections.

2.1 Extracting prototypes

This step is based on the M³ model introduced in Graser et al. (2020). In short, movement data records are clustered into prototypes using an incremental algorithm based on Vector Quantization. In Vector Quantization, probability density functions are modeled by the distribution of so-called prototype vectors. In our approach, these prototypes describe movement properties using Gaussian Mixture Models (GMMs). Each GMM consists of a set of components $\boldsymbol{C}$. Each component c has a set of parameters $\theta_c = \{\boldsymbol{\mu_c}, \boldsymbol{S_c}\}$, where $\boldsymbol{\mu_c}$ is the mean value vector and $\boldsymbol{S_c}$ is the covariance matrix of the multivariate Gaussian. The Leader-Follower clustering (Duda et al., 2001) approach employed adds new data points to the closest existing cluster or creates a new cluster if a specified distance threshold $\mathrm{d_{max}}$ between the data point and the closest cluster is exceeded. To allow for distributed processing, movement data is split using a spatiotemporal grid. The contents of each grid cell are then processed independently.

2.2 Computing flows

After the prototypes have been computed, our new flow algorithm computes transitions between pairs of prototypes. Like the prototypes, our flows are also modeled using GMMs.

The information modeled in each flow includes but is not limited to the number of transitions and the speed distribution. An object moving from prototype A to prototype B triggers an update of the corresponding flow. To allow for distributed processing, each node in the distributed computing environment needs a copy of the previously computed prototypes. Before flows can be computed, movement records are grouped by moving-object ID, sorted chronologically, and then split into trajectories. Each moving object is processed independently. The complete flow algorithm (as illustrated in Figure 2) can be summarized as follows:

1. Create trajectories (i.e. sequences of chronologically ordered records) for individual moving objects:
    a. Split continuous movement tracks at stops and observation gaps and remove outliers
    b. Optionally: generalize the trajectories to reduce data size.
2. For each trajectory:
    a. Let x be the next record in the trajectory
    b. Find the most similar prototype μ^*
    c. Let $\mathrm{d}_{\mathrm{max}}$ be the distance threshold
    d. If $|\mathrm{x} - \mu^*| \leq \mathrm{d}_{\mathrm{max}}$ and μ^* is different from the previous prototype:
        i. Let γ be the flow between μ^* and the previous prototype
        ii. Let α_x, α_γ, α_{μ^*} be the directions of x, γ, and μ^* respectively
        iii. Let $\sigma_{\alpha_{\mu^*}}$ be the standard deviation of the direction of μ^*
        iv. Let φ_{max} be the direction difference threshold
        v. If $|\alpha_x - \alpha_{\mu^*}| \leq \varphi_{max}$ and $|\alpha_x - \alpha_\gamma| \leq \varphi_{max}$ and $|\alpha_x - \alpha_{\mu^*}| \leq 2\sigma_{\alpha_{\mu^*}}$:
            1. Update the flow properties: travel speed and number of transitions
            2. Update the previous prototype reference.

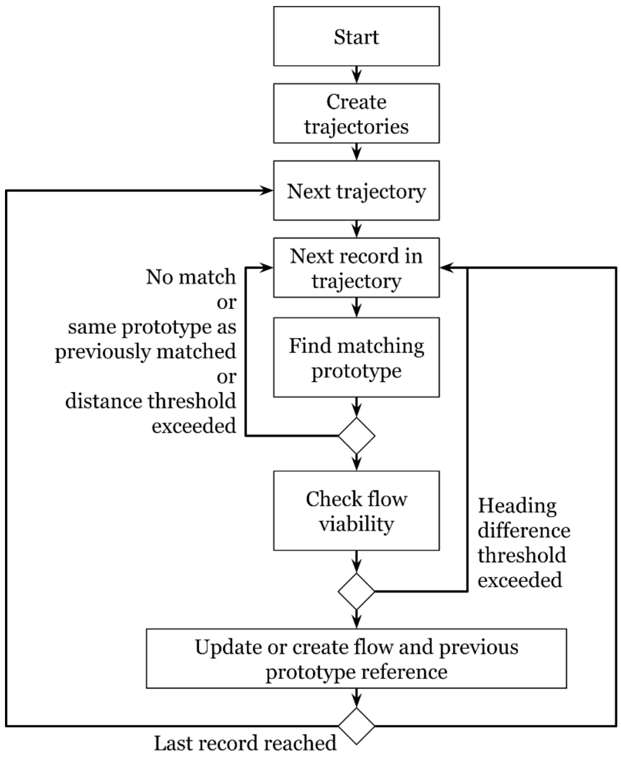
Start
Create trajectories
Next trajectory
Next record in trajectory
Find matching prototype
No match or same prototype as previously matched or distance threshold exceeded
Check flow viability
Heading difference threshold exceeded
Update or create flow and previous prototype reference
Last record reached

Figure 2: Flow diagram for the flow algorithm

Both algorithms for extracting prototypes and computing flows were implemented in Apache Spark. Spark (Zaharia et al., 2010) is a general-purpose cluster-computing framework that supports distributed computing on large datasets which do not fit into the available memory. This is important for processing large movement datasets. For the prototype extraction, only the (intermediate) prototypes and the particular movement record currently being worked on have to be kept in the memory. Similarly, for the flow computations, only the prototypes, the (intermediate) flow results, and the trajectory currently being worked on have to be kept in the memory for each iteration.

3 Case study

This case study aims to extract movement patterns from massive maritime vessel movement data. Vessel movements are tracked by the Automatic Identification System (AIS), which requires that vessels above a certain size broadcast their position and status.

3.1 Input data and cluster setup

The data used in this study were published by the Danish Maritime Authority; our dataset contains 350 million records covering July 2017. To store this data for distributed processing, we use GeoMesa Accumulo. GeoMesa provides fast spatiotemporal indexing (Hughes et al. 2015) to help store and access spatiotemporal data. GeoMesa also provides spatial analysis functions that can be called by Spark.

The computer cluster used in this case study comprises eight data nodes: three nodes with two Intel Xeon E5-2430L CPUs and 32G RAM each, three nodes with two Intel Xeon E5-2660 v3 and 64G RAM each, and two nodes with two Intel Xeon Gold 6136 each. The operating system and HDFS file system reside on SSDs. The setup is based on Apache Hadoop 2.7 and managed using Ambari 2.6.

3.2 Results

Figure 3 and Figure 4 present the resulting prototypes and flows for two different vessel types – passenger and tanker vessels – in the area surrounding Gothenburg, Sweden. Wide flow lines in Figure 3 highlight frequent ferry connections in this area. These connections include local ferries that travel back and forth between the mainland and various islands along the coast, as well as long-distance ferry connections heading towards Denmark and Germany in the south-west.

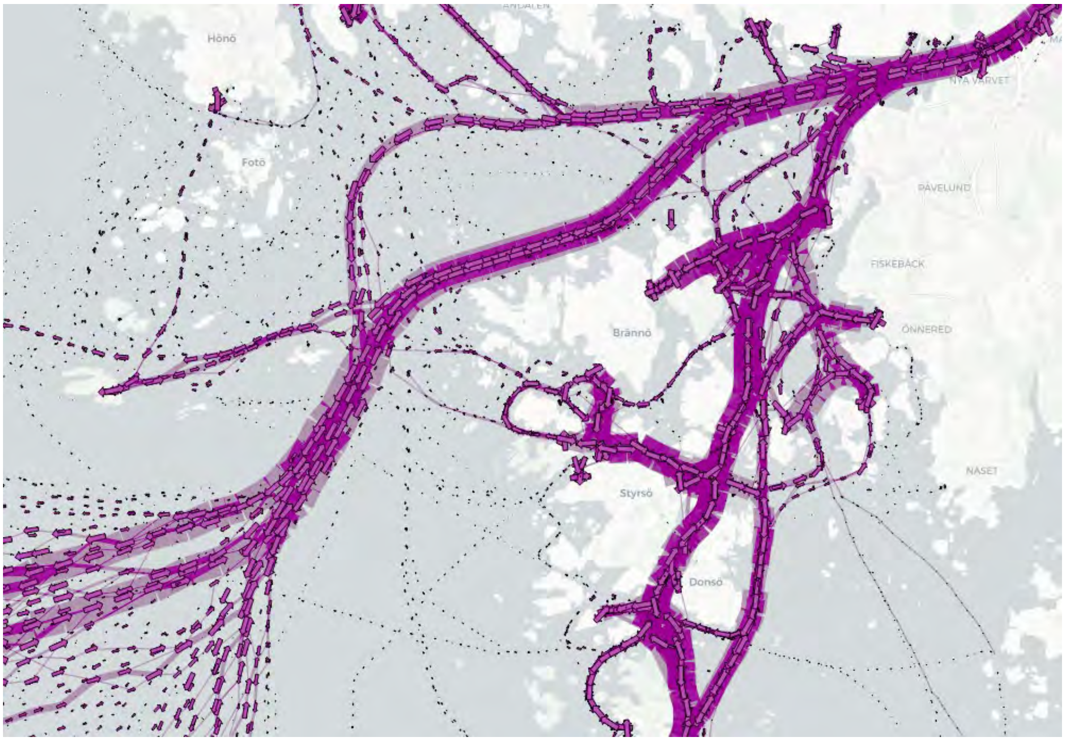

Figure 3: Passenger vessel prototypes (arrows) and flows (connections between arrows)

The tanker flows presented in Figure 4 are mostly focused on the main corridor entering the port of Gothenburg. Smaller flows indicate tankers providing services to islands. Tangles of flow lines and prototypes pointing in various directions in the highlighted region indicate an anchorage area. Indeed, comparisons of these movement patterns and mapped maritime information (Sjöfartsverket, 2016) confirm that this region is a dedicated anchorage area.

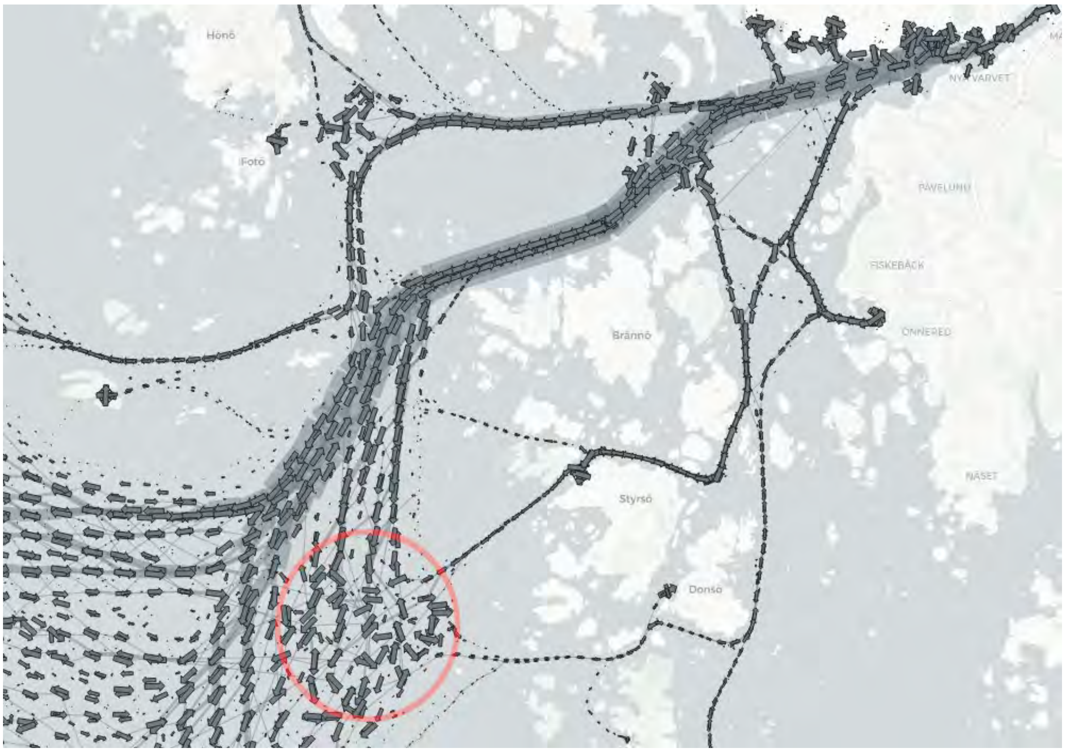

Figure 4: Tanker vessel prototypes (arrows) and flows (connections between arrows). The red circle marks an anchorage area containing tangles of flow lines and prototypes pointing in various directions

Since our flows also include information about mean movement speeds and speed distributions, they also support a more in-depth exploration of speed patterns than regular flow maps (Andrienko & Andrienko, 2011), which model flow strength but not flow speed distribution. For example, Figure 5 shows the speed patterns of passenger vessels. Wide lines indicate a high variation in speed values along a flow. The northern route into and out of the harbor of Gothenburg is particularly noteworthy for its high variations of speed. Information about regions with high speed variation is particularly relevant since these areas need to be watched more closely because accidents are more likely to happen where lots of vessels are moving at different speeds. At its western end, this route splits into darker (higher speed) and lighter (lower speed) routes with lower speed variation. This indicates that vessels with different speed characteristics follow different routes from thereon.

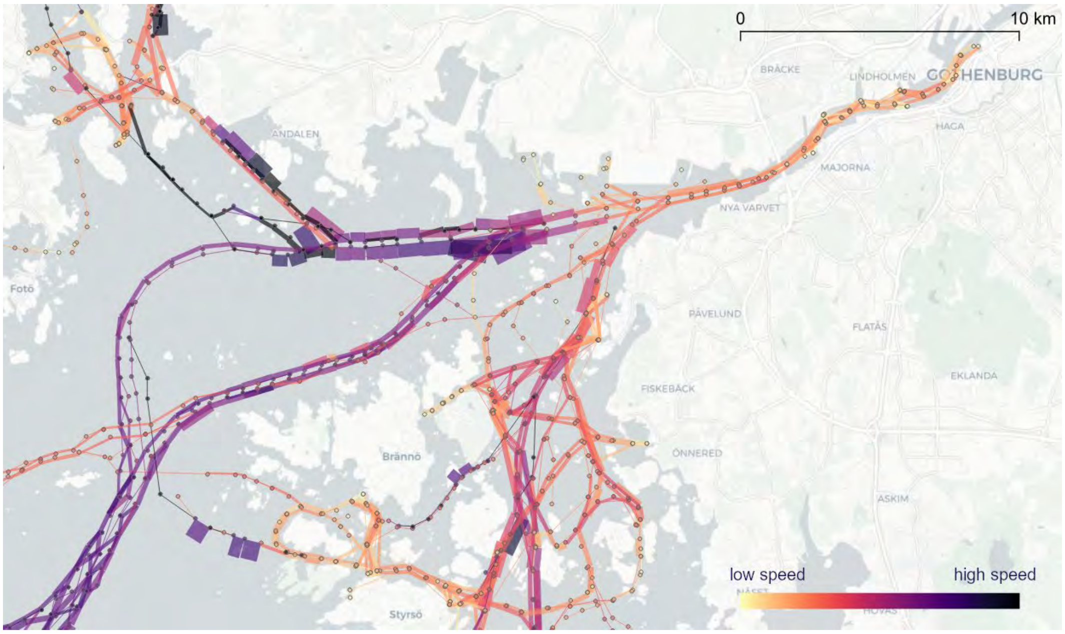

Figure 5: Passenger speed patterns: mean flow speeds (line colour: darker colours equal higher speeds) and speed variation (line width)

As this case study shows, our flows enable the exploration of movement patterns regarding the spatial distribution and density of trajectories, as well as the flow-specific distribution of movement speeds. We discovered, for example, anchorage areas that could be confirmed by official port maps, as well as routes with very high variations in vessel speed which could be important for safety considerations.

4 Discussion

The most critical step in the flow computation is the creation of trajectories. While Spark provides high-level functions for grouping and aggregating records, these are mostly geared towards dealing with unsorted data. If high-level Spark core functionality is used incorrectly, an aggregator needs to collect and sort the entire trajectory in the main memory of a single processing node. Consequently, analyses frequently run into out-of-memory errors when dealing with large datasets. Third-party libraries such as Spark-sorted (Tresata, 2020) provide groupSort, a functionality required to group, sort and iteratively process massive datasets. It never materializes the group for a given key in memory, but instead offers iterator-based streaming of the sorted data. This functionality helps to efficiently build the trajectories which are necessary for computing flows.

The runtime of the computations depends on a variety of factors, including the size of the input dataset, the characteristics of the compute cluster setup (such as the number of Spark executors and their assigned memory), and the spatial resolution of the model (d_{max} and n_{max}

in the prototype extraction step). Models with fewer prototypes and larger cells can be computed faster but provide a less detailed representation of the original observations. A detailed runtime evaluation and sensitivity analysis of the prototype algorithm is provided in Graser et al. (2020). The runtime of the trajectory creation step depends on the efficiency of the groupSort implementation. The runtime of the flow computation step depends on the efficiency of the implementation for finding the matching prototype and thus on the spatial indexing method used.

While the prototype algorithm allows for continuous updates and can therefore handle continuous streams of input data, the flow algorithm does not allow for continuous updates. Flows would have to be recomputed (at least locally) whenever prototypes changed. Therefore, the algorithm does not support exploration of continuous data streams. However, it can be used to explore large historical datasets. To support incremental updates of the flow model, it needs to be integrated into the prototype computation steps. An incremental flow model must keep track of the last observed positions of all moving objects within the system. This introduces considerable memory requirements, since every computational node needs access to this information.

The quality of the flows presented in the case study was assessed using visual plausibility checks. Both the form of the flows (geometries) as well as their strength and speed show expected patterns and are therefore deemed suitable for the exploration of this movement dataset. A quantitative evaluation requires a measure for how well the computed flows represent the original trajectories. To the best of our knowledge, there is no established method that addresses this specific issue. However, measures used to evaluate trajectory generalization algorithms may be adaptable to this issue.

5 Conclusions and future work

We have presented a novel algorithm for extracting flow patterns from large movement datasets. Our new flow algorithm builds on the distributed movement data exploration model M^3 and enables the distributed computation of flows between prototypes. We have demonstrated the usefulness of this approach in a case study involving a large dataset of maritime vessel movements.

While the visualizations in this case study enable a qualitative evaluation of the resulting flows, questions remain pertaining to the quantitative evaluation of the flows. Future work should therefore include the development of quantitative measures that can be used to assess the quality of aggregated flow information.

Potential uses cases for flow data are not limited to data exploration. In the future, we plan to use movement patterns extracted from historical data in predictive analytics, for example, to provide location predictions as well as to estimate time of arrival.

Acknowledgements

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

References

Andrienko, N. & Andrienko, G. (2011). Spatial generalization and aggregation of massive movement data. *IEEE Transactions on visualization and computer graphics*, *17*(2), 205-219.

Andrienko, G., Andrienko, N., Chen, W., Maciejewski, R., & Zhao, Y. (2017a). Visual analytics of mobility and transportation: State of the art and further research directions. *IEEE Transactions on Intelligent Transportation Systems*, *18*(8), 2232-2249.

Andrienko, G., Andrienko, N., Fuchs, G., & Wood, J. (2017b). Revealing patterns and trends of mass mobility through spatial and temporal abstraction of origin-destination movement data. *IEEE transactions on visualization and computer graphics*, *23*(9), 2120-2136. 4

Aronsen, M., & Landmark, K. (2016). Density mapping of ship traffic. FFI-RAPPORT 16/02061. Norwegian Defence Research Establishment (FFI).

Chen, W., Guo, F., & Wang, F. Y. (2015). A survey of traffic data visualization. *IEEE Transactions on Intelligent Transportation Systems*, *16*(6), 2970-2984.

Demšar, U., & Virrantaus, K. (2010). Space–time density of trajectories: exploring spatio-temporal patterns in movement data. *International Journal of Geographical Information Science*, *24*(10), 1527-1542.

Dobrkovic, A., Iacob, M. E., & van Hillegersberg, J. (2018). Maritime pattern extraction and route reconstruction from incomplete AIS data. *International journal of Data science and Analytics*, *5*(2-3), 111-136.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification*. John Wiley & Sons.

Graser, A., Widhalm, P., & Dragaschnig, M. (2020, in print). The M^3 massive movement model: a distributed incrementally updatable solution for big movement data exploration. *International Journal of Geographical Information Science*.

He, J., Chen, H., Chen, Y., Tang, X., & Zou, Y. (2019). Diverse visualization techniques and methods of moving-object-trajectory data: a review. *ISPRS International Journal of Geo-Information*, *8*(2), 63.

Hughes, J. N., Annex, A., Eichelberger, C. N., Fox, A., Hulbert, A., & Ronquest, M. (2015). Geomesa: a distributed architecture for spatio-temporal fusion. In *Geospatial Informatics, Fusion, and Motion Video Analytics V* (Vol. 9473, p. 94730F). International Society for Optics and Photonics.

Miller, H.J. & Goodchild, M.F. (2015). Data-driven geography. *GeoJournal, 80*(4), 449–461.

Pallotta, G., Vespe, M., & Bryan, K. (2013). Vessel pattern knowledge discovery from AIS data: A framework for anomaly detection and route prediction. *Entropy*, *15*(6), 2218-2245.

Rinzivillo, S., Pedreschi, D., Nanni, M., Giannotti, F., Andrienko, N., & Andrienko, G. (2008). Visually driven analysis of movement data by progressive clustering. *Information Visualization*, *7*(3-4), 225-239.

Robinson, A.C., Demšar, U., Moore, A.B., Buckley, A., Jiang, B., Field, K., Kraak, M.J., Camboim, S.P. & Sluter, C.R. (2017). Geospatial big data and cartography: research challenges and opportunities for making maps that matter. *International Journal of Cartography, 3*(1), 32–60.

Sjöfartsverket (2016). Passageplan Göteborg, Retrieved from https://www.sjofartsverket.se/pages/29206/Passageplan%20folder%20Göteborg%202016.pdf

Tresata (2020) Secondary sort and streaming reduce for Apache Spark: tresata/spark-sorted. Retrieved from https://github.com/tresata/spark-sorted

WHO (2018) Global status report on road safety 2018. Technical report, World Health Organization, Geneva. Retrieved from https://apps.who.int/iris/bitstream/handle/10665/276462/9789241565684-eng.pdf

Willems, N., Van De Wetering, H., & Van Wijk, J. J. (2009). Visualization of vessel movements. In *Computer Graphics Forum*, 28(3), pp. 959-966. Oxford, UK: Blackwell.

Xiao, Z., Ponnambalam, L., Fu, X., & Zhang, W. (2017). Maritime traffic probabilistic forecasting based on vessels' waterway patterns and motion behaviors. *IEEE Transactions on Intelligent Transportation Systems*, *18*(11), 3122-3134.

Zaharia, M., Chowdhury, M., Franklin, M.J. et al. (2010). Spark: Cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing,* p. 10.