GPGPU Accelerated Flow Diagrams

Daniel Bird
Daniel.Bird@uea.ac.uk
University Of East Anglia
Norwich, England

Stephen Laycock
S.Laycock@uea.ac.uk
University Of East Anglia
Norwich, England

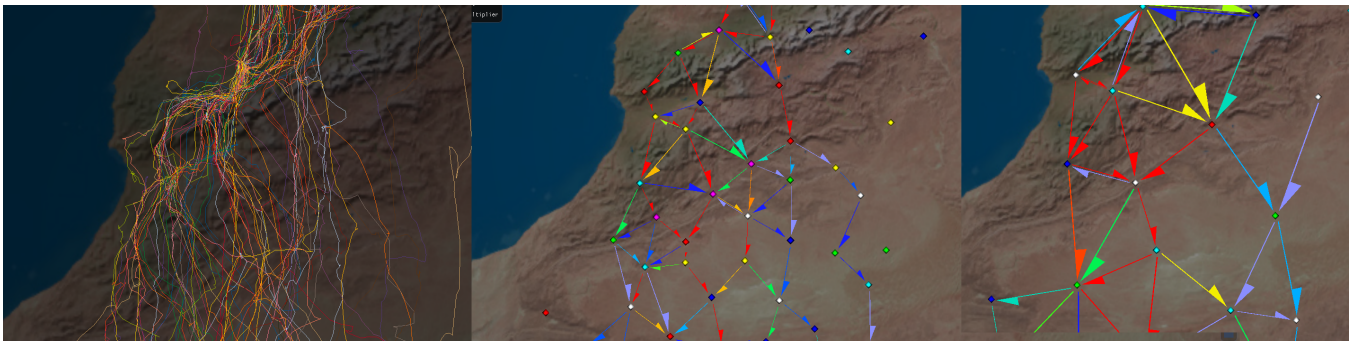

Figure 1: Aggregate flow map of white stork data. First showing the original data, then at generalisation levels of 60km (centre) and 120km (right) The dataset contains over 300,000 points, which was clustered into 318 centroids in 241 milliseconds.

CCS CONCEPTS

• **Human-centered computing** → **Visualization toolkits**; **Visualization techniques**; • **Computing methodologies** → *Parallel algorithms.*

KEYWORDS

Movement Ecology, GPGPU, Visualisation Toolkit

ACM Reference Format:
Daniel Bird and Stephen Laycock. 2021. GPGPU Accelerated Flow Diagrams. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Posters (SIGGRAPH '21 Posters), August 09-13, 2021.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3450618.3469143

1 INTRODUCTION

In today's 'big data' environment the sheer volume of data can cause problems in the creation of visualisations of trajectory data. An overabundance of data creates a number of issues not only tied to the computational complexity of the visualisation, but also with occlusion of data. This is particularly an issue for large datasets, where multiple data points overlap, causing some parts of the data to become obscured. More involved aggregation techniques for large movement datasets of GPS data are now beginning to be developed. This allows for the visualisation of not only where movement is occurring, but also what direction. For example, Andrienko and Andrienko [2011] developed a method of aggregation that creates a number of flows between 'characteristic points'. These characteristic points are defined as relocations within a trajectory of particular interest, such as significant turning points or stopovers. Recent developments by Graser et al. [2020] improved upon this by creating flow maps for large maritime vessel movement datasets. This method however utilises the power of a distributed computing environment. We argue that with adjustments to the original methods used by Andrienko and Andrienko, it can be made more suitable for processing large amounts of data utilising the massively parallel processing of the GPU, without the need for networked clusters.

Here we present a GPGPU accelerated aggregate flow method, used to aggregate the millions of data points found in today's ecological movement datasets. We adapt the sequential method of Andrienko and Andrienko [2011], making it more suitable for the GPU architecture. This allows for interactive adjustment of parameters, removing the disconnect of adjustment and results that occurs when long computation times are required. This creates a fully interactive environment that can be used to investigate large amounts of movement data that can be used to examine phenomena at multiple scales.

2 OUR APPROACH

Much like the method implemented by Andrienko and Andrienko [2011] the creation of aggregate flow diagrams is divided into three distinct steps: The detection of characteristic points, the clustering of these event points to create the centroids used to generalise the data, and finally the segmentation of trajectories into flows. It is assumed here that the characteristic points have already been determined.

To implement clustering a GPGPU accelerated canopy clustering method is used. Originally designed by McCallum et al. [2000] as a pre-clustering algorithm for use with large, high dimensional datasets, its use here allows for the efficient clustering of large

SIGGRAPH '21 Posters, August 09-13, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8371-4/21/08.
https://doi.org/10.1145/3450618.3469143

amounts of data while also adjusting the radius of the resulting clusters, in turn allowing for adjustment of the level of generalisation. There are three inputs: the dataset to be clustered, the loose distance $T1$ and the tight distance $T2$, with $T2 < T1$. This method is implemented on the GPU via the use of thrust, allowing for rapid calculation of clusters with a variable radius, and thus a variable level of generalisation.

Not only the clustering of characteristic points can benefit from GPGPU acceleration, but also the segmentation of trajectories into visits between each cluster. The initial method by Andrienko and Andrienko utilises Delaunay triangulation to create Voronoi cells, with added points to create cells of a more even size and shape. The current cell for each relocation in every single trajectory is then obtained sequentially, with movement descriptors such as time in cell calculated for each relocation.

We make use of dynamic parallelism to accelerate distance measurements, creating a thread for each centroid for each relocation. With the trajectories already stored on the GPU for visualisation, the OpenGL interoperability is used to send the trajectory data to the kernel, which is faster than copying from host memory. A child kernel is then invoked for each relocation within a trajectory. This child kernel performs a parallel reduction on the centroids, calculating the distance from the centroids to the current relocation and reducing to the minimum distance. The resulting array contains the index of the closest centroid for every relocation. This is then used to calculate the number of transitions between each cell. Other statistics, such as length of the path within the cell and time within the cell, can then be calculated using partial reduction on the annotated array.

Stream compaction is performed on this array, with relocations where the next point contains a different centroid being flagged as a flow transition. The flow count is incremented during the stream compaction using atomics. While this may cause issues as the number of flow transitions approaches the size of the trajectory, in most cases the number of flow transitions will be far smaller than the number of relocations. The compacted array is then used to annotate the trajectory with the approaching flow using a scan operation. This annotates the trajectory with the index of the approaching flow, with the final relocations assumed to be part of the previous flow transition.

3 RESULTS

The clustering step has seen an increase in performance due to GPU acceleration. Andrienko and Andrienko state that "the computing time was 265 milliseconds for the 36,328 points" for clustering "54 groups" using their sequential CPU method, with our GPU canopy clustering method able to cluster 299,371 event points in 241 milliseconds creating 318 centroids. This performance improvement increases as more event points are added, especially with datasets with centroids that contain a large number of points. This is due to avoiding the recalculation of centroids for each point added, with the GPU method performing a single parallel reduction.

Trajectory segmentation also benefits from GPU acceleration, with a trajectory of 3 million data points being annotated and flow counts updated in under one second. This calculation time includes all cell calculations of time duration and distance moved. This allows for the creation of multiple flow maps with differing amounts of tag data, highlighting the movement of different studies or a select number of tags within each study. Comparing to the CPU implementation, a trajectory containing 1 million event points requires 20 seconds to annotate, with the GPU implementation still under 1 second. As the number of points increases the speedup observed compared to the CPU method plateaus to about 20 times. With today's datasets containing millions of data points for a single tag this increase in speed significantly effects the total computation time of a complete dataset of migratory animals.

With both tag event detection and level of generalisation significantly effecting the resultant flow map this recalculation may be needed many times. The CPU implementation by Andrienko and Andrienko can have long waiting times between each flow map. This creates a disconnect between parameter adjustment and result, especially as the number of points increases. GPU acceleration significantly reduces the computation time for large datasets, allowing for a multitude of levels of generalisation to be investigated interactively. This can then be combined with environmental context to allow for investigation of the effect of the environment on the movement of large studies [Bird and Laycock 2019].

ACKNOWLEDGMENTS

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. We would also like to thank Dr Aldina Franco for supplying the GPS data for the White Storks [Gilbert et al. 2016] and the discussions on the research.

REFERENCES

Natalia Andrienko and Gennady Andrienko. 2011. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on Visualization and Computer Graphics* 17, 2 (2011), 205–219. https://doi.org/10.1109/TVCG.2010.44

Daniel Bird and Stephen Laycock. 2019. GPGPU Acceleration of Environmental and Movement Datasets. In *ACM SIGGRAPH 2019 Posters* (Los Angeles, California) *(SIGGRAPH '19)*. Association for Computing Machinery, New York, NY, USA, Article 41, 2 pages. https://doi.org/10.1145/3306214.3338584

Nathalie I. Gilbert, Ricardo A. Correia, João Paulo Silva, Carlos Pacheco, Inês Catry, Philip W. Atkinson, Jenny A. Gill, and Aldina M. A. Franco. 2016. Are white storks addicted to junk food? Impacts of landfill use on the movement and behaviour of resident white storks (Ciconia ciconia) from a partially migratory population. *Movement Ecology* 4, 1 (dec 2016), 7. https://doi.org/10.1186/s40462-016-0070-0

Anita Graser, Peter Widhalm, and Melitta Dragaschnig. 2020. The M³ massive movement model: a distributed incrementally updatable solution for big movement data exploration. *International Journal of Geographical Information Science* 34, 12 (dec 2020), 2517–2540. https://doi.org/10.1080/13658816.2020.1776293

Andrew McCallum, Kamal Nigam, and Lyle H Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 169–178.