GPU-Accelerated 3D Visualisation and Analysis of Migratory Behaviour of Long Lived Birds

Daniel Bird

A thesis submitted for the degree of
Doctor of Philosophy
at the University of East Anglia
Dec 2021

GPU-Accelerated 3D Visualisation and Analysis of Migratory Behaviour of Long Lived Birds

Daniel Bird

Abstract

With the amount of data we collect increasing, due to the efficacy of tagging technology improving, the methods we previously applied have begun to take longer and longer to process. As we move forward, it is important that the methods we develop also evolve with the data we collect. Maritime visualisation has already begun to leverage the power of parallel processing to accelerate visualisation. However, some of these techniques require the use of distributed computing, that while useful for datasets that contain billions of points, is harder to implement due to hardware requirements. Here we show that movement ecology can also significantly benefit from the use of parallel processing, while using GPGPU acceleration to enable the use of a single workstation. With only minor adjustments, algorithms can be implemented in parallel, enabling for computation to be completed in real time.

We show this by first implementing a GPGPU accelerated visualisation of global environmental datasets. Through the use of OpenGL and CUDA, it is possible to visualise a dataset containing over 25 million datapoints per timestamp and swap between timestamps in 5ms, allowing for environmental context to be considered when visualising trajectories in real time. These can then be used alongside different GPU accelerated visualisation methods, such as aggregate flow diagrams, to explore large datasets in real time. We also continue to apply GPGPU acceleration to the analysis of migratory data through the use of parallel primitives. With these parallel primitives we show that GPGPU acceleration can allow researchers to accelerate their workflow without the need to completely understand the complexities of GPU programming, allowing for orders of magnitude faster computation times when compared to sequential CPU methods.

Acknowledgements

First and foremost I would like to thank my supervisor Dr Stephen Laycock for their assistance at every stage of this PhD. It can't have been easy to supervise someone who jumps from one idea to the next in the span of a second, but not only did they succeed, their insightful comments and suggestions helped me continue my research through some rather chaotic years. I would also like to thank Dr Aldina Franco, Jethro Gauld, Marta Serra Acacio and many others within the School of Environmental Sciences at UEA for their help guiding development. Without their support, this interdisciplinary research wouldn't be possible. I am also grateful to the NEXUSS CDT for giving me this opportunity to not only study the field in which I am familiar, but also gain a new interest for the environmental sciences. I also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

I would like to express my most sincere gratitude for my friends and family who have supported me. To my cousins James Bird and Racheal Butcher, who inspired me to study from a young age, and my parents, whose constant encouragement and support has made this research possible. Finally, I would like to thank my friend Chris Mudd, whose assistance in writing this thesis has been invaluable.

Contributions

- Publications
  - Daniel Bird and Stephen Laycock. GPGPU acceleration of environmental and movement datasets. In ACM SIGGRAPH 2019 Posters, pages 1–2.2019.
  - Daniel Bird and Stephen Laycock. GPGPU accelerated flow diagrams. In ACM SIGGRAPH 2021 Posters, SIGGRAPH '21, New York, NY, USA,2021. Association for Computing Machinery.
  - Gauld, J. Bird, D. Silva, J. Atkinson, P. Franco, A. Flying in the danger zone: predictive modelling of bird sensitivity to wind farms and power lines. (Manuscript in preparation).
- Seminars, Talks and Q&A
  - Daniel Bird, Ante Qu, Ahmed Mahmoud, Pierre Ecormier-Nocca, Krzysztof Gdawiec, Pooran Memari, Yotam Gingold. SIGGRAPH '21 Summary and Q&A: Structures and Scenery, New York, NY, USA,2021. Association for Computing Machinery.
  - Daniel Bird. (2021, Mar 30). The Global Animal Movement Toolkit. Seminar presented at the Centre for Ecology, Evolution and Conservation, University of East Anglia.
  - Marta Acácio, Inês Catry, Andrea Sorieano-Redondo, Daniel Bird, João Paulo Silva, Phil Atkinson, Aldina M.A. Franco. Portuguese National Ecology Conference 2020. The influence of weather on the migratory performance of juvenile and adult white storks.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The study into the movement of a species, its effects on the fate of individuals and the structure and dynamics of populations has emerged as the field of movement ecology [NGR+08]. With the rise of the 'three Vs' of big data revolutionising the way the world processes and visualises the data we collect [LK15, LO01], new methods of handling these larger datasets in a clearer and informative manner have arisen. Movement ecology is no different, with increased availability and accuracy of GPS sensors, the field has been able to benefit from larger datasets of movement data. An example of one such device is shown within Figure 1.1. As miniaturisation and efficacy of electronic components improves additional sensors can begin to be coupled with the GPS tracking to enable features related to the animal's state at a given position to be recorded, resulting in movement datasets that contain millions of relocations. This has led to the rise of new research fields, such as visual analytics [AAD+10, AA13, SBvLP+12] and geovisual analytics [Rob17, ÇGS+20], and there is now extensive literature on how we can visualise this new influx of data. This has led to the development of a number of toolsets and R packages that allow for the visualisation of spatio-temporal data, such as DynamoVis [XD15] and MoveVis [SRSW20]. These methods, however, begin to struggle as the ever increasing amount of data we

Figure 1.1: White Stork with logger attached (Picture taken by Aldina Franco).

collect begins to make these methods untenable. For example, Space time cubes become too noisy and difficult to interpret [AA07] and sequential CPU based methods can require long computation times when dealing with large amounts of data. While there has been research into modifying existing visualisation methods to make them easier to interpret [SWS+19, WSN19], the processing times required are still a barrier when processing the millions of data points that are now becoming common place in movement dataset repositories [Mro18]. Recent research has begun to leverage the power of GPGPU acceleration to address the issue of long computation times when processing movement data [HLZL20].

General Purpose Graphics Processing Unit (GPGPU) acceleration has been applied to a growing number of fields to allow for larger datasets to be processed faster than standard sequential CPU methods [WK13, FH15, SDL, KWZ19]. The massively parallel methods that become possible via GPU computing lend themselves well to the processing of spatio-temporal data. We see, however, that a large amount of current literature for movement ecology is either processing large amounts of data either

through sequential methods using the CPU [PBS+18], or relying on online services [DBW+13]. While there are methods that utilise networked clusters to process large amounts of data [GWD20], we believe that there is a middle ground to be achieved via the use of GPGPU computing. With single trajectories now containing millions of datapoints as standard, it is important to use the correct methods for the scale of processing we are hoping to achieve. There is little advantage to be gained by applying parallel processing methods to a dataset that only contains a thousand datapoints, likewise waiting days for the sequential processing of a dataset containing billions of datapoints.

Within this thesis we present a middle ground between the methods that use networked clusters, and the standard sequential methods developed and available via the ever growing repositories of R packages. This middle ground would allow for a researcher to process millions of datapoints in real-time, using only their single workstation. This would enable the processing of more data than CPU based methods, while not requiring the specialised hardware of distributed computing. Through the use of GPGPU computing, we demonstrate that by changing the way we approach a particular problem, and optimising for parallel throughput, many commonly used methods within movement ecology can be performed in real-time.

1.1 Thesis Outline

In Chapter 3, we show this by creating a GPGPU accelerated visualisation of global environmental variables. Literature has shown that the surrounding environmental variables have a significant effect on the movement of species [DBC+15]. With popular methods within literature, if a user wants to quickly create a visualisation of their trajectory data, they can either only visualise the values at each relocation [XD15] or render their visualisation on the CPU which may take a significant amount of time

[SRSW20]. With the application of GPGPU acceleration, we show that it is possible to visualise the millions of datapoints per timestamp that is standard in today's global environmental datasets in real time (5ms). We show that even with lower performance GPUs, there is still a significant increase in performance compared to CPU methods. With real-time rendering of global environmental variables, alongside trajectory data, we show that researchers can investigate their data interactively.

In Chapter 4, with movement data visualised alongside global environmental variables, we apply GPGPU acceleration to the analysis of movement data. As datasets continue to grow and data driven methods become more common [WTB+20], scalability of methods used becomes especially important. This becomes particularly apparent for the analysis techniques that are sensitive to parameter adjustment, especially so when initially investigating data through data driven methods, where ideal parameters are unknown. While there are R packages to process these datasets on the CPU, these can begin to take a significant amount of time to process [SFA19]. Through the application of GPGPU acceleration, we have shown that it is possible to calculate most commonly used metrics within movement ecology in real-time. These metrics can then be processed in the identification of event points, relocations within a trajectory that have been identified as being significant in decision making, such as turning angles and migration start and end points. We have shown that these can be calculated interactively, allowing for the rapid recalculation of an analysis when exploring movement data, removing the disconnect between starting the analysis and the results. These can then be combined with the visualisation of global environmental datasets to allow for multiple variables, such as step speed, tuning angles and net squared displacement, to be visualised simultaneously with environmental factors.

While being able to display large amounts of data can be useful, overplotting of data can cause significant issues when attempting to interpret visualisations [GG13].

This is a common issue with space-time cubes [AA07]. In Chapter 5 we implement a spatial generalisation method by Andrienko and Andrienko [AA11] using the power of GPGPU acceleration. These aggregate flow maps allow for the investigation of large amounts of data, while minimising the issues of overplotting. The application of GPGPU computing allows for the rapid recalculation of the clustering step, resulting in faster adjustments to the level of generalisation when creating flow maps, allowing for researchers to investigate their data at multiple differing scales. This is then combined with the visualisation of global environmental data from Chapter 3, with a multitude of different visualisation methods available. With GPGPU acceleration applied to each visualisation method it becomes possible to create an interactive visualisation large amounts of trajectory data in three distinct manners: The global view that allows for an interactive globe with large amounts of environmental data to be visualised alongside the trajectory, The detail view that allows for closer inspection of trajectory data with an included digital elevation model and the filtered space time cube, a novel combination of the traditional space time cube combined with the aggregation benefits of flow maps that allows for multiple levels of generalisation to be investigated in quick succession interactively.

Literature has observed that during the development of interdisciplinary research, communication between researchers of differing disciplines is key, especially so for toolset creation [RRFH20, Cam05]. The research presented here is no different. Throughout development we have worked with the School of Environmental Science at the University of East Anglia to obtain feedback and guide development. To this end, Chapter 6 shows the development of an R package allowing for researchers to integrate the toolkit and methods presented within this thesis in their research. We show that the toolkit is able to both import and export trajectory data, while also allowing for ease of access of global environmental datasets via the Copernicus Data

Store (CDS) [MS19]. We then show a case study co-authored with Jethro Gauld. Here, we use methods described throughout this thesis to create an input to a generalised linear model that can be used to predict if a bird is flying at a danger height for renewable energy sources and power lines. The initial results of this are then visualised using the methods described throughout this thesis.

To facilitate the use of GPGPU accelerated computing within movement ecology, the methods presented here have been collated into a single visual analytics toolkit; the Global Animal Movement Toolkit (GAMT). It is hoped that this visual analytics toolkit will allow for researchers to leverage the power of GPU hardware within movement ecology, with the newfound interactivity allowing for exploration and analysis of large amounts of movement data with environmental context in real-time.

Chapter 2

Related Work

Cartography has long since been an area of study, with arrows, points, colours and printed text being used to visualise and investigate different spatial processes [RWER05, KK92, Sie84, KO20, Cor01, CP]. These early maps were not only used as a method of navigation, but also communication, allowing for people to understand the world around them. Early advancements into maps as a visualisation medium created famous maps such as John Snow's 1854 map of cholera deaths in London [Tul18] and Charles Picquet's heat map of cholera cases in 1832, one of the first uses of geovisualisation in epidemiology [CP].

The advancement of computer graphics and database technology had a drastic impact on this domain, separating "digital" or "computer" cartography from "traditional" cartography. The ability to display, filter and query maps through menus and legends on a computer interface changed the way we looked at spatial data [KO20]. This changed maps from the final product they used to be to an interface capable of investigating spatial phenomena, moving away from the static interpretation of the world around us to a dynamic one. This was the advent of the electronic atlas [Sie84]. The ability to animate this data allowed for the visualisation of changes over time, without the need for multiple static maps [KK92].

It was around the 1990s that the term geovisualisation began to be used as a

way to distinguish digital cartography from traditional cartography, replacing digital cartography [CJF18, ÇBAD17]. The study of geovisualisation, shortened from geographic visualisation, has been defined as the study and exploration of spatial data facilitated by the interactivity of digital maps [Kra03a]. With the technological advancements occurring in the 1990s, conceptual frameworks were also proposed to facilitate the definition of geovisualisation. A significantly prevalent framework was MacEachren's Cartography^3 [Mac94]. This framework not only conceptualised the users and the task, but also the level of interactivity available. This framework has had multiple redefinitions [KO20, MGP$^+$04] with more recent definitions showing the four main situations to visualise spatial data, to present, synthesise, analyse and explore. This is shown within Figure 2.1.

2.1 The Rise of Geographic Information Systems (GIS)

With the increased technical capabilities of computer hardware, users were able to query and analyse spatial data using specialised software. This led to the development of Geographic Information Systems (GIS), frameworks for analysing and querying spatial data. One of the earliest and possibly most influential of these systems was Roger Tomlinson's Canada Geographic Information System (CGIS) [Goo18], which was developed during the 1960s. Roger Tomlinson is now known as "the modern father of GIS" for this early work and coining of the term geographic information system. CGIS continued private development until the 1990s and was never publicly available.

As technology continued to improve, multiple GIS began development, with commercially available GIS solutions growing in number. In 1981 Esri released one of the first commercially available GIS systems ARC/INFO. This would later be updated

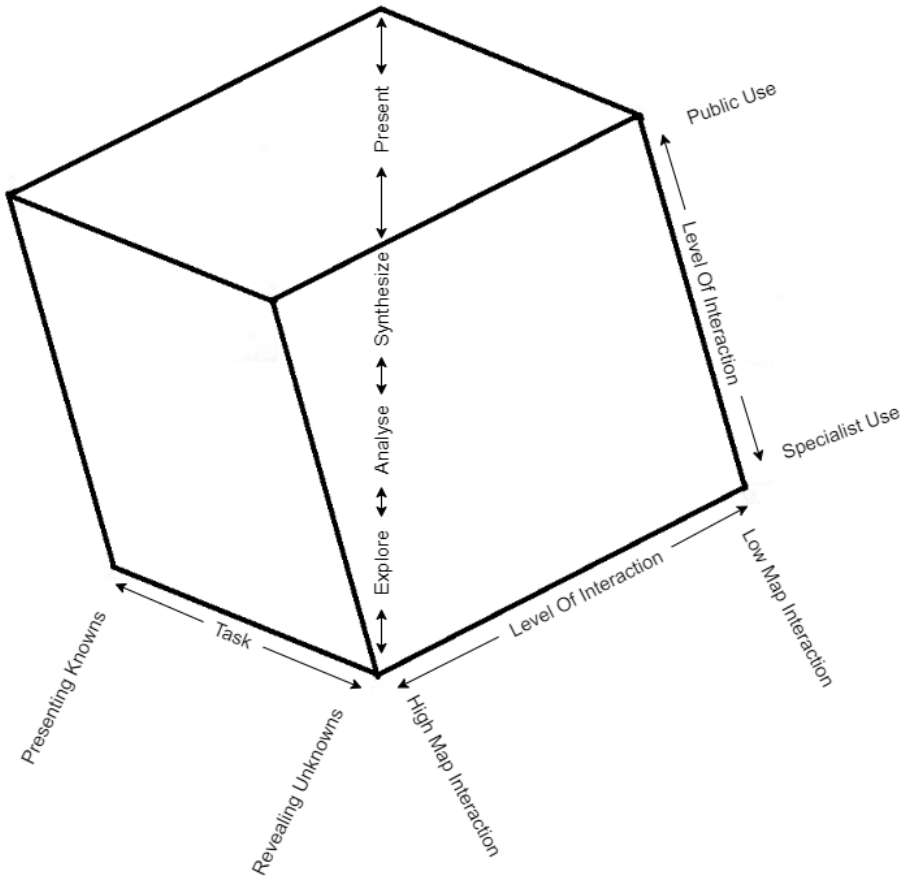


Figure 2.1: The Cartography^3 framework, redrawn from MacEachren [Mac94]. This framework describes the core functions for geovisualisation and GIS.

to ArcMaps and ArcGIS, one of the most popular GIS solutions today [SJ10]. These commercially available solutions however suffered from a few pitfalls, they tended to be architecturally closed systems that required specialised technical skills to effectively operate. This has recently led to the development of open source GIS, such as CommonGIS, QGIS and GRASS GIS have seen continuous development for a number of different platforms and programming languages [QGI19, AAV03, NM13]. It should be noted however that the GIS solutions today "provide only limited time dimension support" [GD20]. This has in turn led to little movement data visualisation and analysis solutions as standard in most GIS.

2.2 Map Projections, Geodesy and Geodesic Distances

As our understanding of the world around us increased, our methods of mapping it evolved. Due to the nature of attempting to represent the round earth on a flat surface, such as a screen or map, a projection must be applied. These projections have changed significantly over time, with old projections being changed to suit a new purpose and new projections being developed to overcome the downsides of others. No one map projection can perfectly represent the globe as a whole, with advantages and disadvantages to each projection. Due to this, care should be taken when selecting a map projection, ensuring that the projection matches its intended use. For example, initially designed for navigational use, the Mercator projection is the most widely recognisable projection in cartography [Sny87]. As a cylindrical based map projection, the Mercator projection preserves local directions and shapes and is thus conformal, aiding in navigation. This projection comes with a number of often overlooked downfalls however, it significantly distorts at higher latitudes, with area unable to be visually compared [Jen12]. A common example of this is the comparison of Greenland and South America, with the former only being one-eighth the size of the later despite their similar size using the Mercator projection. This also results in the map becoming nearly unusable at the poles, severely distorting any areas or trajectories. In 2005, with Google maps quickly becoming the leading mapping service, the web Mercator projection became the standard for online maps. This projection is similar to the standard Mercator projection but differs in that it is mapped to a sphere instead of a cylinder, thus the web Mercator is also known as the spherical Mercator [BFUY14]. It is this projection that is used by most online mapping services (Google maps, ArcGIS Online and Bing maps) due to its square shape for the entire world when it is truncated at 85°, allowing for square buildings

and streets to maintain their 90° angles and to allow for equal size square tiles for mapping APIs [Goo]. This projection however still has the significant distortions of the standard Mercator projection, it cannot be used for area based visualisations, such as dot density maps, and regular tessellation cannot be used for binning when placed over a standard Mercator projection due to the area within each bin differing, especially at the poles.

Due to this, when using a map for visualisation or analysis, care should be taken when selecting a projection, as it can have a significant effect on the interpretation of the results. While the nature of mapping the globe to a flat surface requires some form of compromise, due to the sheer number of projections now available there is a projection available for most uses. A number of factors should be considered when selecting a projection: preservation of shape, preservation of area, preservation of scale and preservation of direction, with a number of projections that compromise between these properties also being developed [SJWS15, ŠPJ19]. Examples of some common projections are shown within Figure 2.2. It should be noted that Savric et al. performed a study to determine the preferred projection of map users and discovered that the cartographic knowledge of the individual had an effect on the preferred projection [SJWS15], with the Mercator projection the least preferred for professionals, but remaining slightly higher for general map readers.

When performing studies that require spatial data, especially over large areas, it is important to accurately represent the spatial position of an object. Geodesy is the study of representing and measuring the geometry of the earth and other planets. Due to the earths geometric shape not being a perfect sphere, but instead an imperfect ellipsoid with differing levels of elevation, it is important to define a coordinate system that can accurately represent spatial positions. To facilitate this, a number of different Geographic Coordinate Systems (GCS), often incorrectly referred

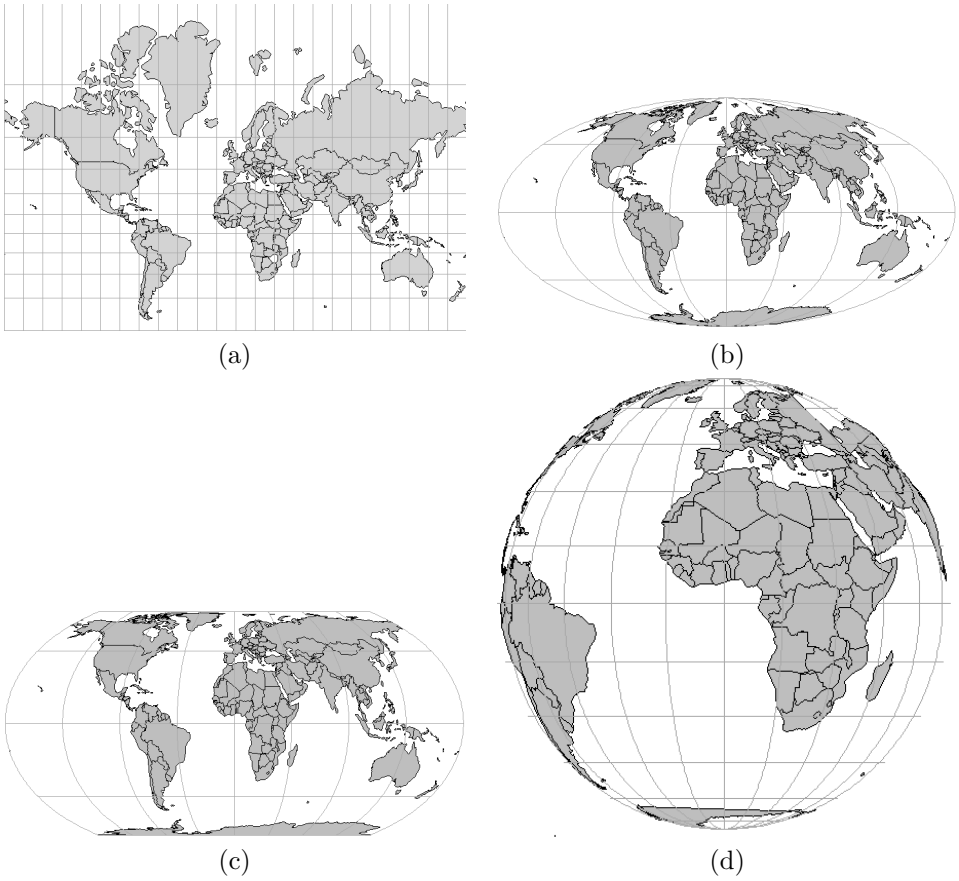


Figure 2.2: Examples of different map projections. The Mercator projection (a), the Mollweide projection [GH95] (b), the Equal Earth Map Projection [ŠPJ19] (c) and the orthographic projection (d).

to as datums [SM98], are used depending on the location of the study. A GCS is made up of three components: an angular unit of measure, a prime meridian and a reference spheroid (datum). The most commonly used GCS is the World Geodetic System (WGS), the latest revision of which is WGS 84. With the coordinate origin of WGS 84 being defined at the earths centre of mass, WGS 84 defines the surface of the earth to be an oblate spheroid [SM98] and is used as the coordinate system for most global applications, such as GPS [WRM12]. This GCS is not the only one in use however, a large number of local datums that more closely fit a smaller area of the earths surface are in use [Esr12], but are unsuitable for visualisation that require

global use. A common method of converting between different GCS is the use of the Earth Cantered Earth Fixed (ECEF) geocentric coordinate system. ECEF is defined as a Cartesian coordinate system that has the Earths centre of mass as the origin, with a point being defined as (X, Y, Z) geocentric coordinates, instead of the geodetic coordinates of longitude (λ), latitude (ϕ) and height (h) [ZLB99]. This is shown within Figure 2.3. Due to the large number of differing coordinate systems, projections and datums used, care should be taken to only compare data when they are in the same coordinate system. This led to ArcGIS and other GIS automatically re-projecting any point and spatial data to the same GCS automatically, ensuring the user is working in the same GCS for all data.

Calculating the distance between two points is important for most spatial analysis. While some studies disregard the curved nature of the earth, due to only working with small areas where its effects are negligible [AA11], as the distance increases the inaccuracies increase dramatically, and in these cases geodesic distances are used. A Geodesic line is defined as "the shortest path between two points on a curved surface" [Ket14]. Two of the most common ways to determine the distance of two geodetic coordinates are the Haversine method [Sin84] and Vincenty's Formulae [Vin75]. The haversine formulae determines the great arc distance d between two points $P_1 = (\lambda_1, \phi_1), P_2 = (\lambda_2, \phi_2)$ on a sphere, and is calculated as follows:

$$a = \sin^2(\frac{\phi_2 - \phi_1}{2}) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2(\frac{\lambda_2 - \lambda_1}{2}) \tag{2.2.1}$$

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a}) \tag{2.2.2}$$

$$d = R \cdot c \tag{2.2.3}$$

With R being the radius of the sphere. In the case of geodesic distances this is often

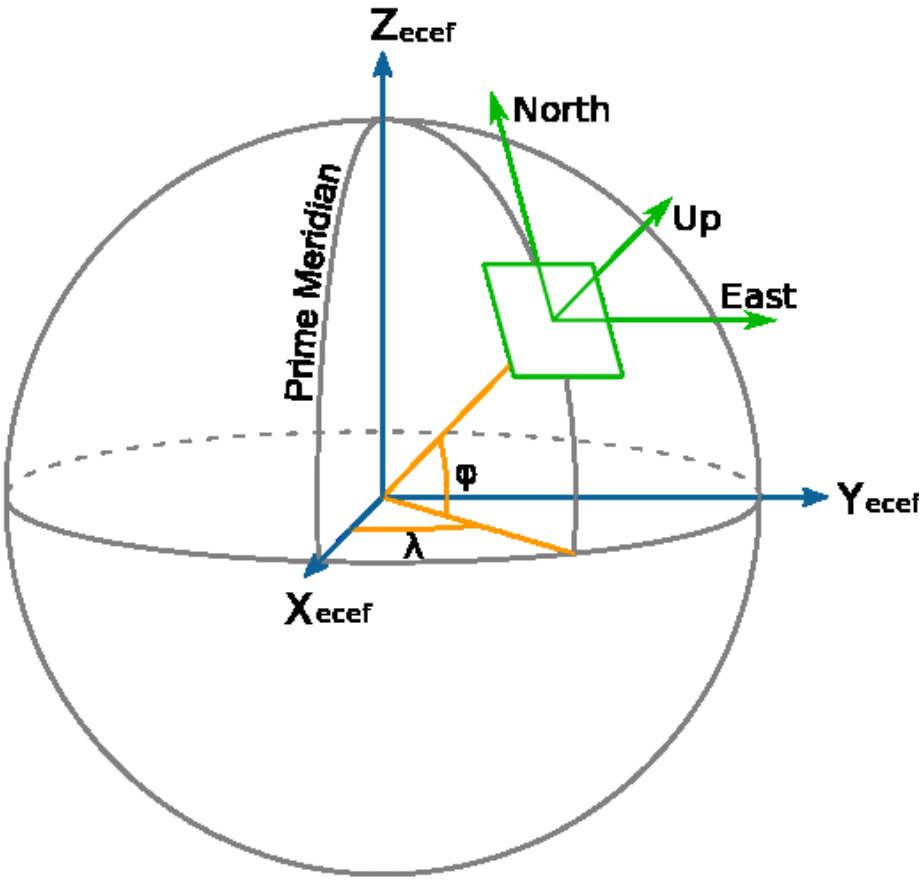


Figure 2.3: The ECEF coordinate system, with the X axis passing through the equator at 0° longitude, the y axis passing though the equator at 90° longitude and the z axis passing though the north pole.

the mean earth radius of 6371.009km. Due to the assumption that earth is a sphere, rather an ellipsoid, an error of up to 0.5% is observed. While this is accurate enough for most applications, especially those over smaller distances, Vincenty's Formulae accounts for this. Vincenty's formulae were devised by Thaddeus Vincenty in 1975 to calculate the distance between two points on the surface of a spheroid. The first of the formulae is the direct method that computes the location of a new point given a starting point and a direction (or azimuth). The second of the formulae, known as the inverse method, calculates the geographical distance between two points on a spheroid. Both of these formulae are iterative methods, requiring the convergence of

a variable to a desired accuracy. Due to the involved nature of the maths involved more information can be found in [Vin75, Rap93]. It should be noted that while the accuracy of Vincenty is better, with an accuracy about 0.5mm [Ket17, Vin75], this iterative method does have its downfalls, with antipolar points requiring a large number of iterations to converge to a solution. This iterative solution also makes it slightly unsuitable for GPGPU usage, as different points will require variable amounts of iteration, causing warp divergence. Warp divergence and its effect on computation time are discussed in Section 2.7.

2.3 The advancement of big Data

With the rise of the internet, the miniaturisation and increased capabilities of sensors, and database technology becoming more advanced in the last few decades, there has been an explosion of available data. This sudden abundance of data has had a profound effect on a number of sectors, such as retail [BGKV17], economics [MP19], government and policy making [HHKP19], geospatial data [LK15] and many more. Due to this rapid evolution, big data has been redefined numerous times by many authors. Many authors agree that big data is characterised by three main components: volume, velocity and variety, also known as "the three Vs". [LO01].

Volume describes the large volume of data being output from the abundance of devices and sensors that are prevalent in today's society. Sagiroglu and Sinanc state that "5 exabytes of data were created until 2003"[SS13].

Velocity describes not only the rate at which big data is collected, but also the need for new technologies to rapidly process and interpret the large amount of data as it is being obtained. For example, in 2020 YouTube had "500 hours of video uploaded every minute" [Woj20].

Variety describes the multitude of sources and formats that big data originates,

from raw financial data to different video formats, and the availability of both structured and unstructured data as well as private, public and incomplete datasets [KCN15].

Recently authors have begun to add other 'Vs' to this definition. For example, Gandomi and Haider define veracity as "the unreliability inherent in some sources of data" [GH15]. Emani et al. also argue value is a defining characteristic of big data, that the data obtained has a value that can be extracted [KCN15].

GIS systems have recently been following this trend of utilising big data for geospatial analysis [WZW19, SKSŚ17]. For example, ArcGIS utilises Hadoop, an open source toolkit for spatial analysis of big data, to allow users to leverage large amounts of data.

Movement ecology has also begun to benefit from the evolution of big data. Data repositories that contain millions of datapoints are now becoming available to researchers. For example, the Dryad digital repository was created to host data for journals in evolution and ecology and allow for the sharing of data between researchers to "promote data intensive knowledge discovery by providing open, persistent, robust and secure access to well described and easily discovered data about life on earth and its physical environment" [Vis10]. This in turn led to the development of Movebank. Movebank is an online repository of movement data specifically for animal movement ecology. It began in 2007 and has since increased in size in the last decade. As of 2021 Movebank contains 2.6 billion relocations, with a further 3.2 billion non-location events that make up 6,202 studies of 1,056 different taxa [Mro18]. Movebank also has a number of tools associated to allow for the management of live feeds of tracing hardware and the processing of datasets. The Env-DATA system (Environmental Data Automated Track Annotation System) is a commonly used tool that allows for environmental data from a multitude of predetermined sources including MODIS, ASTER and ETOP01 to be appended to movement data, allowing for the

investigation of environmental context and its effect on animal movement.

Gorodov et al. have observed that the nature of big data has been shown to present problems during visualisation [GG13], which they classify into five main issues: Visual noise, large image perception, information Loss, high performance cost, and high rate of image change. Visual noise describes the overlapping and noisiness that occurs when large amounts of data are plotted together. Large image perception describes the attempted solution to the previous problem where the resolution of visualisations increase in an attempt to stop the overlapping of datapoints. This however results in images too large to acquire any useful information. Information loss describes the issues surrounding data aggregation, where as we simplify our data through generalisation, we loose much of the information associated with our data. High performance requirements denotes the significant amount of processing required for large datasets. Finally, high rate of image change describes the difficulty in monitoring large amounts of data for changes. When designing new visualisation methods for large amounts of data, it is important to take these into consideration.

2.4 Collection of Movement Data and Biotelemetry in Movement Ecology

Movement ecology has seen a renewed interest in recent years due to the influx of new biotelemetry methods [WNB+15]. Where previously researchers would struggle to investigate basic theories, for example relying on attached balloons in the case of marine animals [RH09] and kitchen timers to measure diving capabilities [Koo65], new methods of obtaining telemetry data have allowed access to near real time tracking of animals possible [UCD+10]. Biotelemetry, defined as "the remote determination of an animals status" [Rod01, Pri92], has evolved rapidly since the initial use telemetry data.

One of the earliest methods of obtaining biotelemetry data was the use of Very High Frequency (VHF) radio transmitters. These VHF transmitters were attached to animals, often referred to as 'tags' throughout literature, and pulsed radio signals were then received by a nearby receiver, which could then be used to infer the position of the tracked animal by either "homing in" on the radio signal via the use of directional antenna or using multiple receivers to triangulate the position [Rod01]. These early VHF systems suffered from a number of problems, the most prevalent of which was the range of the transmitters. While the range of these transmitters could reach up to 20km [Rod01] large scale movement, such as migratory data, moved much further than this. This lack of range was further exacerbated by the surrounding environment, such as thick vegetation reducing range to less than 100m [MJLW02]. To alleviate these range issues, VHF tracking was improved to include the use of aircraft and satellites in orbit to allow for wider coverage. In particular, the French space agency (CNES), the National Aeronautics and Space Administration (NASA) and the US National Oceanic and Atmospheric Administration (NOAA) created the CLS/service Argos Data Collection and Location System [Cla89] that has been used to track multiple species since the 1980s [MCNF92, WG12]. It is commonly referred to as the Argos system throughout literature. The Argos system works via the radio transmission of data from Platform Transmitter Terminals (PTTs) attached to animals, with positional data being calculated using Doppler shift [Cla89].

Concurrent to the development of the Argos system was the development of the Global Positioning System (GPS). Initially developed by the US military under the name Navstar GPS, GPS was designed as a replacement to previous navigation systems to allow for 24-hour accurate positional data. This works via the positioning of 24 satellites in orbit, with the position of the ith satellite being (X_i, Y_i, Z_i). At

all times at least 4 satellites are in view from any position on earth. The users position (U_x, U_y, U_z) can then be inferred using P_i, the distance from the ith satellite to the users position, which is calculated from the time required for a signal from the satellite to reach the user ΔT_B [RMAM14].

$$(X_i - U_x)^2 + (Y_i - U_y)^2 + (Z_i - U_z)^2 = (P_i - c\Delta T_B)^2. \tag{2.4.1}$$

In March 1990 the selective availability of GPS was implemented. This intentional degradation of the GPS system by the US military reserved the most accurate positioning system for military use. SA reduced the precision of non-military positional systems to approximately 100m. This was later suspended and then finally removed in 2007. Following the removal of SA GPS locations were shown to be within 10-28m [HR08]. The precision of the GPS system is still affected by outside influence. Different environmental variables have been shown to have a varied effect on precision, which are still being investigated. When assessing the efficacy of GPS sensors in differing environments two metrics are commonly used throughout ligature, fix rate reduction and measurement error [FFH$^+$10]. To measure error, a GPS tag is placed within an environment affixed at varying heights to simulate the species desired and a regular sampling interval of GPS data is obtained. The fix rate is calculated by obtaining the number of successful recorded locations, known as fixes, and dividing by the number of attempts to obtain fixes. The precision of GPS data can also be compared by calculating the distance from the recorded location and the true location, this is referred to as the measurement error. This is a common way of testing GPS accuracy and is known as a stationary test. These stationary tests allow for easier comparison of results and ease of replication across a number of differing environmental conditions [FFH$^+$10, DSK05, IKJM05]. Mobile device tests have been performed to attempt to measure the effect of differing types of movement but are

rarer [FFH+10].

Canopy closure has been shown to have a significant effect on both the fix rate and measurement error as the amount of canopy closure increases [DSK05]. The topography of the surrounding terrain has also been shown to effect both fix rate and measurement error. Cain et al. assessed the effect of topography by calculating the amount of 'measurable sky', defined as the amount of sky that is visible from a location in all directions and angles [IKJM05]. It was shown that as the amount of available sky decreased, the fix rate and measurement errors were increased. The surrounding environment is not the only variable in the precision of GPS, the position and the number of satellites used to calculate the fix have been shown to have an effect on precision [LRGV07]. When a GPS device records its current position, the number of satellites and their relative position can be stored. A GPS location can either have a two dimensional solution that was calculated using three satellites, or a three dimensional solution that uses four or more satellites. It has been shown that the two dimensional solutions tend to have a precision of 36m or less, while three dimensional solutions have a higher precision of 12m or less [LRGV07, FFH+10]. The relative position of the satellites used also have an effect of the precision of the location recorded. Dilution Of Precision (DOP) is an index calculated from the distance between the satellites used to obtain the position of a GPS receiver. The closer the satellites are to each other, the higher the DOP, with lower DOP values representing a wider spacing. DOP can be expressed as a number of differing measurements, such as PDOP which has been used to apply DOP based screening of points to reduce the amount of erroneous records within a trajectory [RR97, LRGV07].

In comparison, GPS tags have been found to be a more accurate than other technologies such as VHF [TFKB10, RH09]. This newfound precision and high sampling rates alongside global coverage have allowed for new datasets larger than ever before.

With a new influx of data, ecologists have now been able to investigate a number of different ecological phenomena and answer different ecological questions, such as habitat selection [ORJ15, FSS+17], animal movement [OSFM10, SAF+20] and foraging behaviours [GCS+16a].

2.5 Visualisation Of Spatio-temporal data

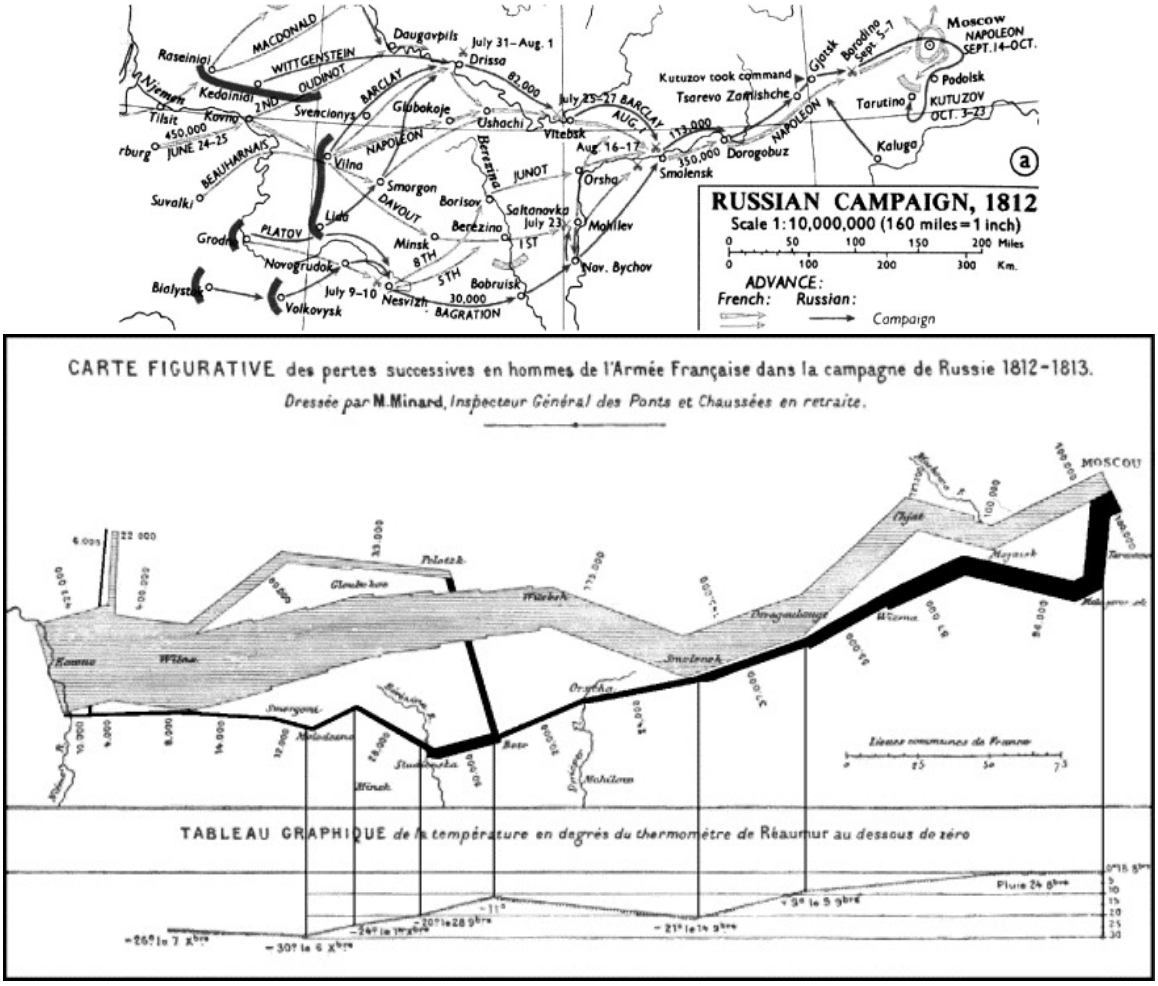


Figure 2.4: Napoleons Russian Campaign of 1812. First shown is Darby and Fullard's map, showing troop movement in full detail using lines arrows and printed dates [DF70]. Below shown is Charles Minard's visualisation, showing position, troop numbers, temperature and time [Cor01].

Visualisation of spatio-temporal data has a long and detailed literature, with

many methods of visualising spatio-temporal patterns, such as arrows, points, lines and printed dates being suggested [DF70, ZWZM12]. As can be seen from Figure 2.5 these visualisations can quickly become overly complex, leading to a difficulty in comprehension. Early developments were made in the efficient and clear visualisation of spatial data. Figure 2.4 shows Charles Minard's map of Napoleon's march on Moscow [Cor01]. This map allows for the visualisation of multiple distinct variables of longitude, latitude, temperature, troop numbers and time, receiving praise as "the best statistical graphic ever drawn" [Tuf85]. Charles Minard's work with flow maps also allowed for the visualisation of multiple spatiotemporal processes, such as road traffic between Dijon and Mulhouse [Ren18]. This trend of improvement continued, with Hägerstrand creating the space time cube in 1967 [H$^+$68]. The space time cube allowed for the visualisation of both the spatial and temporal aspects of trajectories. This was achieved by plotting the longitude and latitude of the current position on the x and y axis respectively, then plotting the current time on the z axis. These first space time cubes were drawn by hand, restricting the visualisation to only a single point of view.

The advancement of computer graphics also had a significant impact on the visualisation of spatiotemporal data. The previously static and hand drawn visualisations of trajectories could now be interactive, with animations able to dynamically visualise the temporal component of trajectory data. This led to other interdisciplinary fields of research that shared much in common. Visual analytics emerged from the field of information visualisation and attempts to combine the processing power of computers with human interaction with a single interface to gleam insights into the multitude of data now available [CT05]. This in turn led to geovisual analytics. Geovisual analytics diverges from geovisualisation in "its explicit inclusion of computational methods" [CJF18]. While recent visual analytics research has been

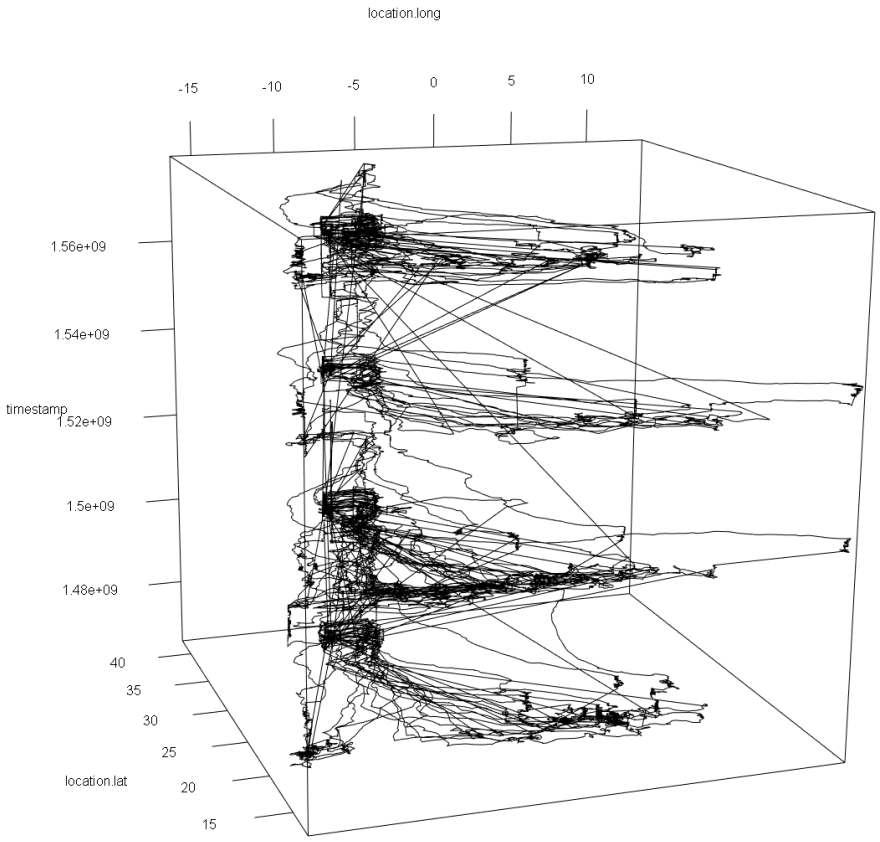


Figure 2.5: Space Time Cube of the migrations of 30 white storks [FCRA19] over three years of migrations, created using R. As can be seen, as large amounts of data begin to be plotted the visualisation becomes unusable.

performed on spatio-temporal data from social media [CLY17, DAA+19, AAC+17b] there have also been developments in movement ecology and maritime visualisation [GWD20, SWS+19, SRSW20, HLZL20].

2.5.1 Recent Developments in Visualisation of Movement Data

Recent surveys have been conducted in the field of visualisation for movement ecology, and have highlighted that toolsets investigating movement within context is important [DLBP+21, NGR+08]. Demšar et al [DLBP+21] have also observed that while different disciplines have their own approaches to trajectory analysis and visualisation, the underlying concepts are the same, with interdisciplinary collaborations allowing for advances to be made. Due to this multidisciplinary nature of movement ecology, multiple researchers of differing backgrounds are involved, with some researchers having little experience with visualisation of geophysical datasets, leading to the design of visualisation tools that allow for researchers to visualise their data.

One such implementation is the rworldmap R package [Sou11]. Designed by South. A, it allows users to visualise their global datasets with any colour mapping. rworldmap allows for the visualisation of either gridded or country datasets and the ability to export them as a static plot. This limits the research capabilities of the visualisation by not being able to visualise the changes of the dataset over time without multiple plots. The ability to visualise changes in a dataset over time would allow for investigation into these changes effect movement of tagged species.

MoveVis [SRSW20] is an R package that allows users to visualise movement datasets over either satellite imagery or environmental datasets. Implemented via the use of the ggplots package, moveVis creates each frame on the CPU and then exports either a GIF or video file of the entire visualisation. While creating small scale animations using the CPU may be feasible, creating longer animations with

global environmental datasets requires significant time, with the animation having to be recreated when adjustments are made. The computation time required for an animation duration of 20 seconds requires "approximately 25–35 minutes". These long computation times for a relatively short animation require a long stop gap between input of parameters and final result. Being unable to interact with the datasets loaded the ability to recognise and investigate movement patterns is limited. Due to being implemented entirely on the CPU the scalability of this visualisation package is also limited, requiring even longer computation times as the range of the visualisation increases.

Xavier and Dodge [XD15] also created dynamoVis, using the java programming environment, for the visualisation of Movebank [Mro18]. DynamoVis allowed for users to apply colour filters to any data attached to the locations recorded, enabling visualisation of any data that was also logged by the tag, such as external temperature. DynamoVis in combination with ENV-Data allows researchers to investigate and visualise their data. While integration of environmental context and interactivity was achieved, it is unable to show the environmental variables within the surrounding area of a recorded point. This lack of visualisation of surrounding environmental variables limits the researcher's ability to identify the possible effect environmental variables have on movement decisions.

With the advancement of computer graphics space times cubes have seen a resurgence in interest. Kraak began to apply interactive computer graphics to allow for interactive rendering of space time cubes, allowing for the visualisation and analysis of individual or group movement [Kra03b]. This later drew criticism from Andrienko and Andrienko [AA07] due to over complication and occlusion issues as the number of trajectories within the space time cube increased. To combat this, multiple solutions were suggested. Andrienko and Andrienko suggested the reduction of these

trajectories into m-events, characteristic points that describe significant points within a trajectory, which can then be plotted within a space time cube [AAH+13]. Recent studies with the space time cube include the use of AR or VR to allow for interactivity with a space time cube [SWS+19, WSN19]. According to their user studies, the amount of errors during investigation were reduced, but are limited by their hardware requirements.

The advantages and disadvantages of 3D space time cubes and 2D animated spatio-temporal data have recently been investigated. Mullaw investigated the advantages of space time cubes in comparison to animations and static maps and discovered that animation performed better in most cases, with the exception of the identification of stopovers and returns to a previous path [Dem08]. Amini et al. [ARH+15] continued to investigate the advantages of interactivity in both 2D animated trajectories and 3D space time cubes. Participants in the study were asked to interpret a set of data using either an animated 2D or 3D representation and asked to answer questions based on the data. They discovered that the 3D representation significantly reduced the answering time for complex questions. They also conclude on how the combination of 2D and 3D representations through seamless interactions could provide improvement to current methods.

2.5.2 Aggregation of Movement Data and Flow Maps

With the recent abundance of data in all sectors there has been an increased development into the methods to visualise large amounts of movement data. Both standard animations and space time cubes suffer from occlusion when attempting to render large amounts of data. Space time cubes in particular create messy visualisations with an overabundance of data. As a result of this, methods of aggregation have been explored in an attempt to allow for visualisations to employ as much of the data as

possible while still allowing for meaningful visualisation.

A prevalent method of aggregation is the use of density surfaces (2D) or density areas (3D) to reduce the visual clutter of standard trajectories. These methods aggregate movement data by creating a histogram of the entire surface (or area in the 3D case) and applying a colour map to highlight areas where movement occurs the most. Demšar and Virrantaus apply a 3D linear kernel function to each trajectory to create a density area of use for maritime data [DV10a]. This reduced visual clutter when working with the exceptionally large maritime datasets and allowed for the visualisation of space use over time. This approach has a drawback however, while the visualisation preserves the space use over time when used in a space time cube it no longer reveals directional information or speed. This approach instead lends itself to the investigation of space utilisation instead of explicit movement visualisation [NT20].

Flow maps have also recently seen development as a method of reducing visual clutter. Flow maps do not represent the exact trajectories of data, but instead typically represent the aggregate movement between two nodes, also described as "origin destination pairs" [ÇGS+20], with the number of trajectories that move between these two nodes typically represented by either colour or thickness of the flow itself [JSM+18]. Recent developments of flow maps include the use of 3D curvature to allow for flow maps in virtual reality, reducing the amount of occlusion of typical 2D methods [YDJ+19].

To allow for the visualisation of massive movement data spatial aggregation techniques have been applied to trajectories to allow for clearer visualisation. Andrienko N. and Andrienko G. created a method of aggregation using the java programming environment [AA11]. First, 'characteristic points' within the trajectory are detected.

These points are significant events within a trajectory that describe meaningful or insightful events within the trajectory, such as stopovers, turning points, start and end points. This is achieved by sequentially processing each point. With these characteristic points detected they are then grouped in space, creating clusters of a desired radius. Andrienko N. and Andrienko G. considered the use of multiple clustering methods to achieve the desired convex spatial clusters with a desired radius, thus allowing for a desired level of generalisation. They determined k-means and k-medioids inappropriate as they require a known number of centroids, and other methods such as DBSCAN [EKSX96] allowed for density-based clustering but not of a desired radius. This led to the development of a clustering method that utilises a regular grid to cluster points.

The clustering method used by Andrienko N. and Andrienko G. [AA11] takes two parameters, the set of characteristic points and a desired maximum radius. This radius dictates the level generalisation, the larger the radius the more generalised the resultant flow map. The study area is first divided into a regular grid, with each grid cell having the width and height equal to the max radius. The characteristic points are then sequentially added to the grid. The use of a regular grid means that only the nine grid cells containing and surrounding the point need to be checked for clusters. The point is then added to the closest cluster within the max radius, the centroid of that cluster is then recalculated with the additional point. If no centroids are found within the nine grid cells or there is no cluster within the max radius a new cluster is created with the new point. This is repeated until there are no new points within the dataset. With this completed, a set of Voronoi cells are created using the centroids, with additional cells added to areas without any centroids to ensure more even sized cells. Each trajectory is then sequentially processed to segment the trajectories into transitions between these cells, it is the number of these transitions that determine

the size of the resulting visualised flow.

Recent developments by Graser et al. also create flow maps, but instead of relying on the hardware of a single workstation they make use of a networked cluster [GWD20]. Named the M^3 movement model, Graser et al utilise Gaussian Mixture Models (GMMs) to aggregate movement data while also using a flexible number of data driven prototypes within a regular grid. This allows for more detail representation without the need to adjust the regular grid to an impractical resolution. This method utilises the processing power and parallel potential of networked clusters to aggregate and visualise exceptionally large amounts of data. For example, Graser et al. create an aggregate flow diagram of 3.9 billion records in 41 minutes.

2.6 Analysis of Animal Movement Data

The miniaturisation and improvement in accuracy of GPS sensors has led to an influx of available data, with the big data revolution and repositories like Movebank allowing researchers to access more data than ever before. This has led to rapid development of analytical techniques in movement ecology, in turn leading to frameworks to unify movement research. One of the most impactful is the framework proposed by Nathana et al [NGR+08]. This framework identifies four fundamental questions of animal movement: Why move? How to move? Where and when to move? and What are the consequences of this movement? To answer these questions Nathana et al. created a general conceptual framework for movement ecology, shown in Figure 2.6. This framework separates movement into differing components: the external factors of movement, such as environmental variables; the navigation capacity of the individual, the traits that enables an individual to orientate itself within space/time; the internal state, the state of the individual made up of a multitude of physiological and neurological states that effect the individual's ability and readiness to move; the

motion capacity, the individuals ability to execute or facilitate movement which results in the movement path, the resultant movement due to the previous components. This unifying framework has led to a number of developments in movement research for both animal movement ecology and human mobility research.

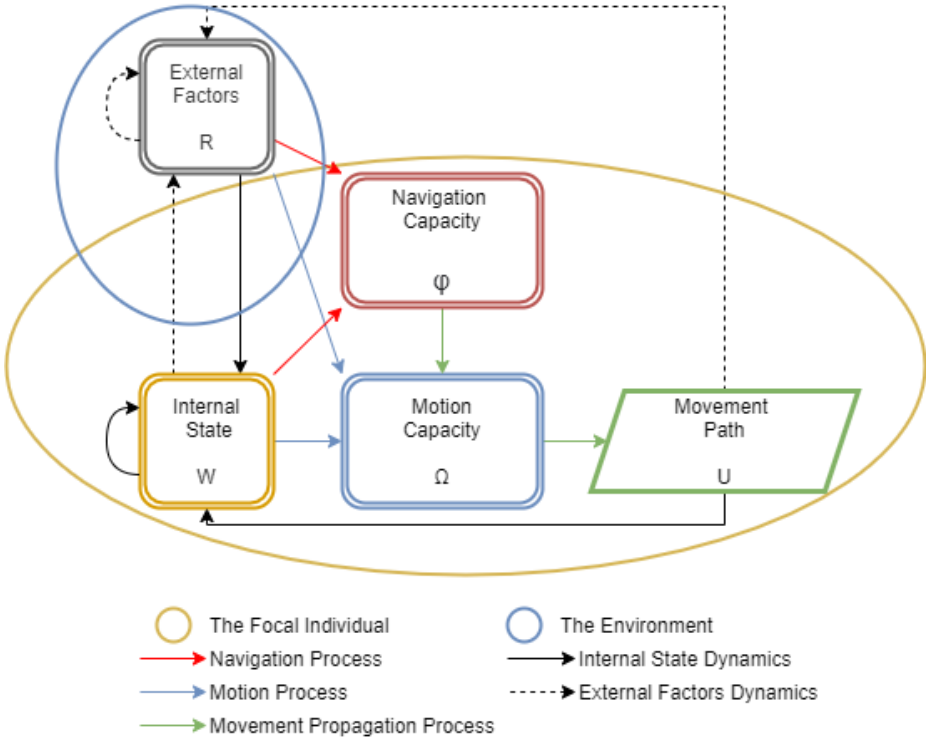


Figure 2.6: The redrawn movement ecology framework conceptualised by Nathan el al [NGR+08]. The focal individual is described in terms of the internal state, the motion capacity and navigation capacity (yellow ellipsoid) and the External factors dictated by the environment (blue ellipsoid). Each of these components are linked via processes that effect each component.

Recently, Williams et al. have also created a framework to unify research biologging sensor data called the integrated Biologging Framework [WTB+20]. This framework highlights the interdisciplinary nature of biologging technology and the data it creates, highlighting that due to this complex and multifaceted nature interdisciplinary collaborations are key for optimal study design. Williams et al. state four areas critical to study design: Questions, Sensors, Data and Analysis. The order

in which these are considered will change depending on either a question/hypothesis or data driven route, referred to as pathways.

2.6.1 Movement Descriptors and Metrics for Animal Movement Ecology

Even with the abundance of tracking data now available it is unusual to observe the continuous, complete path of a tracked animal. As an alternative a discrete number of observations along the path of the tracked animal is observed [CDRC09, GS08, NGR+08]. These recorded relocations, ordered by the time of their creation, result in a movement track or trajectory. The accuracy of this trajectory can be influenced by a number of factors, such as sampling interval, the spatial accuracy of technology used (GPS, VHF) and irregular sampling frequencies caused by either an intentional decision to prolong the longevity of onboard sensors [BLK+12] or from device malfunction and poor signal [FSF+18].

A common early step for the investigation of trajectory data is the calculation of movement descriptors and metrics for path-level analysis [SDCG18]. These metrics, also known as path-signals [ESB16] or stepwise characteristics [SDCG18], are typically split into two classifications: Primary and secondary metrics [SDCG18]. Primary metrics consist of descriptors that are directly derived from each relocation within a trajectory. Due to this nature, primary metrics are highly sensitive to the sampling interval of the underlying trajectory.

Primary metrics are not exclusive to the position of the tagged animal. As the miniaturisation of sensors allow for more complex tagging data to be obtained other metrics are also available. For example, accelerometers can be used in the calculation of energy expenditure and have been used to provide an understanding of the smaller scale movements of species, especially at a foraging level where the efficiency of energy expenditure has an impact on survival. The calculation of energy expenditure has

been investigated thoroughly throughout literature and is of significant importance for research. While there are several methods of calculating energy expenditure, such as doubly labelled water and heartbeat rate, these methods are either of low temporal resolution or require invasive methods. Overall Body Dynamic Acceleration calculates a proxy of energy expenditure using triaxial accelerometer data [WWQ+06]. ODBA is calculated by summing the Dynamic component of the accelerometer data, as shown in Equation 2.6.1, with the three axis recorded by the accelerometer denoted by A_x, A_y and A_z. There is one possible problem with this solution however, it assumes that acceleration is not a vector. To correct this a vectoral approach was suggested by Gleiss, Wilson et al. [GWS11]. This approach instead calculated the Vectoral Dynamic Body Acceleration (VeDBA). This equation is shown within Equation 2.6.2.

$$ODBA = |A_x| + |A_y| + |A_z| \tag{2.6.1}$$

$$VeDBA = \sqrt{A_x^2 + A_y^2 + A_z^2} \tag{2.6.2}$$

Research into the comparison between VeDBA and ODBA has shown that while theoretically VeDBA should be a more accurate proxy for energy expenditure, they show very similar results. It was noticed however that VeDBA is a better estimation in the rare case that consistent device orientation could not be guaranteed [QCW+12].

Secondary metrics include summary statistics that are calculated from primary metrics or from a coarser temporal scale than the underlying trajectory data [SDCG18]. These metrics are used extensively in literature to facilitate the characterisation of movement and changing of states within a trajectory. For example, net squared displacement (NSD) is a metric that can be utilised for the identification of movement strategies and patterns of an individual over time [BRPY+15a]. This is achieved by

calculating the squared Euclidean distance between the starting location of an individual's movement and each subsequent relocation along the trajectory. This can begin to show common characteristics entailing to the type of movement of the individual, with literature often classifying movement into four distinct movement patterns; sedentarism, dispersal, nomadism, and migration [BRPY$^+$15a, SAE16]. NSD has recently been used in tandem with non-linear mixed models to identify the beginning and end of migrations, as well as classify movement modes of species [SAE16].

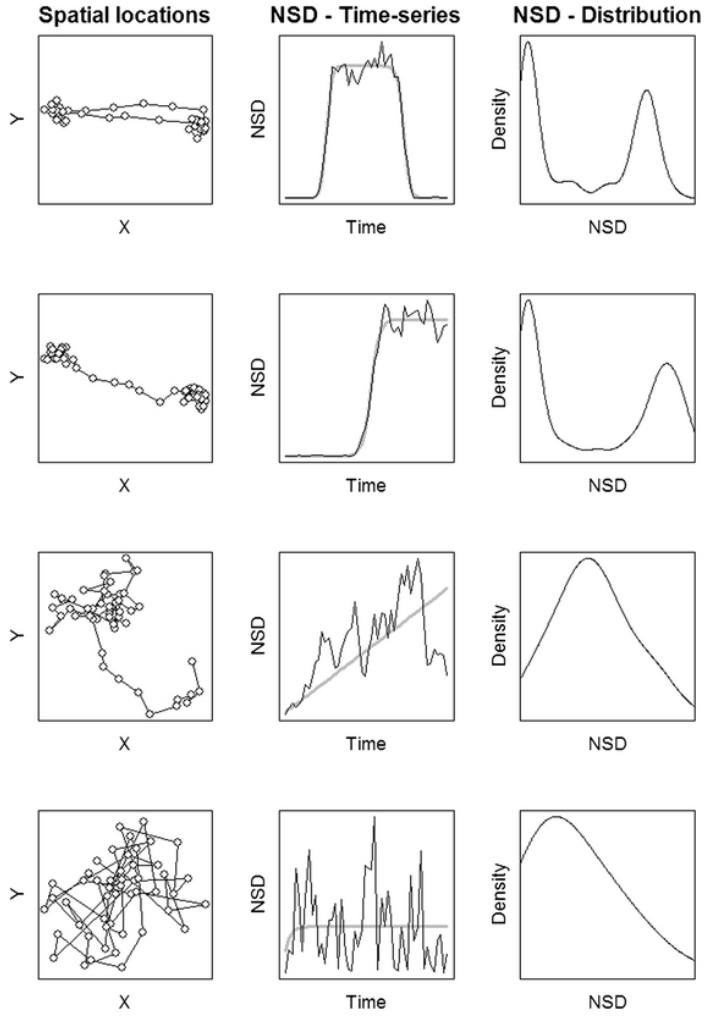


Figure 2.7: The different movement strategies that become apparent with the use of Net Squared Displacement: migration, dispersal, nomadism and sedentarism [BRPY$^+$15a].

First Passage time is another commonly used secondary metric that has been used to differentiate between movement behaviours in animals [FT03, LCDC14]. It is calculated by determining the time required to leave a certain radius. This metric has been used by Webber et al. as a proxy for the amount of foraging occurring, stating "Higher first-passage time reflects a slower linear movement rate … and presumably more intensive local foraging. By contrast, lower first-passage time reflects a faster linear movement rate and presumably less intensive local foraging" [WLB+20].

2.6.2 Segmentation, Event Detection and Behavioural Classification in Movement Data

The classification of behavioural modes and detection of events and characteristic points within a trajectory has seen a significant rise in research due to the abundance of data now being collected [ESB16]. These data streams not only use positional data but also data from other onboard sensors, such as accelerometer data. These metrics can then be used to infer the behavioural state of the animal at a given time. Many methods to achieve this have been utilised in literature [Bat81, ME02, PBS+18, CDRC09, DWL08, SAF+20]. These methods may include the use of specialised knowledge to threshold primary or secondary signals, such as pressure data, to identify when an animal has dived underwater [Koo65].

More advanced methods of classification have seen use in recent literature, such as the use of machine learning techniques. These include the use of KNN clustering techniques to classify behavioural modes [BCGL+14], Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) to segment movement data into different movement modes [SWQ+08, NSFR+12]. Other statistical methods have also been used to process movement data, such as Hidden Markov Models (HMMs) and State Space Modelling (SMM) for classification [PBBG09]. While these complex models allow for accurate classification of movement, the merits of simpler methods are still

apparent [CVB+20]. Williams et al. state that "The convenience of machine-learning systems is that they require little specialist knowledge and information about the data streams from the researcher" [WTB+20]. This lack of specialist knowledge could lead to erroneous use of well established methods. Studies comparing the efficacy of these complex methods to more simplistic methods have found that simpler models may outperform more complex ones in certain situations [SAF+20, CVB+20].

Identification of turning angles, and thus possible decision points, is an ongoing field of research. Recent developments by Potts et al [PBS+18] utilise the high resolution turning data of modern tags to identify significant turning regions in high resolution movement data. The primary metric of heading is used to calculate the Squared Circular Standard Deviation (SCSD) at each relocation using a sliding window. Let $h_1, ..., h_n$ be the heading for each relocation within a trajectory, obtained with a regular sampling interval at time points $t_1, ..., t_n$. A window size of W is used about each relocation to calculate the SCSD s_i of each reallocation using the following equations.

$$s_i = \ln(\frac{1}{\bar{R}_i^2}) \tag{2.6.3}$$

$$\bar{R}_i = \sqrt{\bar{\sin}(h_i)^2 + \bar{\cos}(h_i)^2} \tag{2.6.4}$$

Where $\bar{\sin}(h_i)$ is the average of $\sin(h_i)$ over the window surrounding the relocation and $\bar{\cos}(h_i)$ is the average of $\cos(h_i)$ over the same points. With the SCSD calculated for each relocation spikes above this value may be flagged as turning points. These flagged turning points can then be further filtered to include only those that are above a desired threshold.

2.6.3 Resampling of Movement Data

Due to the often unreliable nature of movement data collection a number of papers investigating the resampling of trajectories to regular sampling intervals and the calculation of missing timestamps have been published. These methods differ in their methods due to the large variety of differing types of movement, with no one method being more accurate in all cases than another [Lon16]. First, we define movement as a set of relocations along a path with given time intervals $z(t)$. An unknown relocation that occurs between two known relocations is defined as $z(t_u)$, with the two known relocations defined as $z(t_i)$ and $z(t_j)$, with $t_i < t_u < t_j$.

A common method due to its simplicity is linear interpolation [SPFCC17, PDGM$^+$21, WHR$^+$21, NLLS15]. This method assumes that the unknown movement between two locations follows a straight path (beeline) at a constant speed and is calculated as follows:

$$z(t_u) = z(t_i) + \frac{t_u - t_i}{t_j - t_i}(z(t_j) - z(t_i)) \tag{2.6.5}$$

While this method is fast and simple to implement, it inaccurately assumes that previous kinematic state of the trajectory has no impact on future relocations. Technologies utilising adaptive sampling however have used linear interpolation effectively, with linear resampling having little effect on the geometry of the resultant trajectory [WHR$^+$21].

Random walks are an interpolation method that utilise random samples from two distributions, the distribution of step lengths and the distribution of turning angles. Originally designed to describe the irregular motion of particles [CPB08] they have since been adjusted for use in ecology. These methods include Continuous-Time Correlated Random Walks (CTCRW) [JLLD08], Drifted Random Walk (DRW) [CBT10] and the Constrained Random Walk (CRW) [TOSW15]. The constrained

random walk algorithm accounts for the sequential nature of movement by using space time prisms as a way of constraining the possible relocations. The space time prisms are then calculated using an upper bound on mobility v_{max}.

Spline interpolation methods have also been used throughout literature, with Catmull-rom splines [YSK11], Bezier Curves [BBB95] and cubic Hermite splines [LS73] being used for ecological applications [HPB10, PFG+21, TSF+06, DWA+16]. Cubic Bezier curves have been shown to be effective in interpolating the movement of animal data within a fluid environment [TSF+06]. Bezier curves are calculated using four control points, two of which are the known relocations $z(t_i)$ and $z(t_j)$ and two of which control the curvature P_2 and P_3. These control points can be obtained using the kinematic state at each of the known relocations [Lon16].

$$P_2 = z(t_i) + v(t_i)\frac{1}{2}(t_j - t_i) \tag{2.6.6}$$

$$P_3 = z(t_j) - v(t_j)\frac{1}{2}(t_j - t_i) \tag{2.6.7}$$

Catmull-Rom spline interpolation is a cubic spline that uses the directly observed relocations as control points [YSK11]. Instead of inferring the control points using the kinematic state at the points $z(t_i)$ and $z(t_j)$, the previous and subsequent points $z(t_{i-1})$ and $z(t_{j+1})$ are used. These four points can then be used to interpolate the position of $z(t_u)$. Hermite splines have also been used to interpolate maritime data [HPB10]. This method uses the two known relocations as control points in tandem with the control parameters of speed and heading.

Long J. implemented a method of interpolation of movement data, based on object kinematics [Lon16]. By using the velocity at the two known relocations and solving for a linear function of acceleration, the position and velocity at point $z(t_u)$ can be inferred. This method is extended to two dimensions by solving the linear function

of acceleration in each dimension independently. The resampling and interpolation of movement data is an ongoing field of research, with new methods utilising new splines being proposed [CBM+21].

2.7 GPGPU Acceleration

Initially, Graphics Processing Units (GPUs) were used as a hardware accelerator for 3D graphics applications. These computationally complex tasks required substantial parallelism to allow for interactive applications, and thus resulted in the development of high performance, high throughput streaming processors. While initially used exclusively for graphics applications, there were early attempts to utilise this graphics pipeline for non-graphics applications. These were successful, but required intimate computer graphics knowledge, requiring complex mapping of program data to the available shader buffers. The term General Purpose Graphics Processing Units (GPGPU) was coined by Mark J. Harris in the early 2000s [HLG+] and from this early success led to GPU vendors creating unified shader architectures that could be more easily programmable [LNOM08]. This later led to the development and release of APIs that utilised higher level abstractions to facilitate the use of GPGPU programming [Kir07, Mun09]. For example NVIDIA's CUDA [Kir07], originally standing for Compute Unified Device Architecture, allowed for the programming of NVIDIA cards for general purpose computing, with AMD also releasing their AMD CTM (Close to Metal). This was followed by OpenCL, a framework for GPGPU computing regardless of GPU vendor [Mun09]. These frameworks allowed for a number of heterogeneous computing systems that utilise both GPGPUs alongside CPUs to accelerate computation.

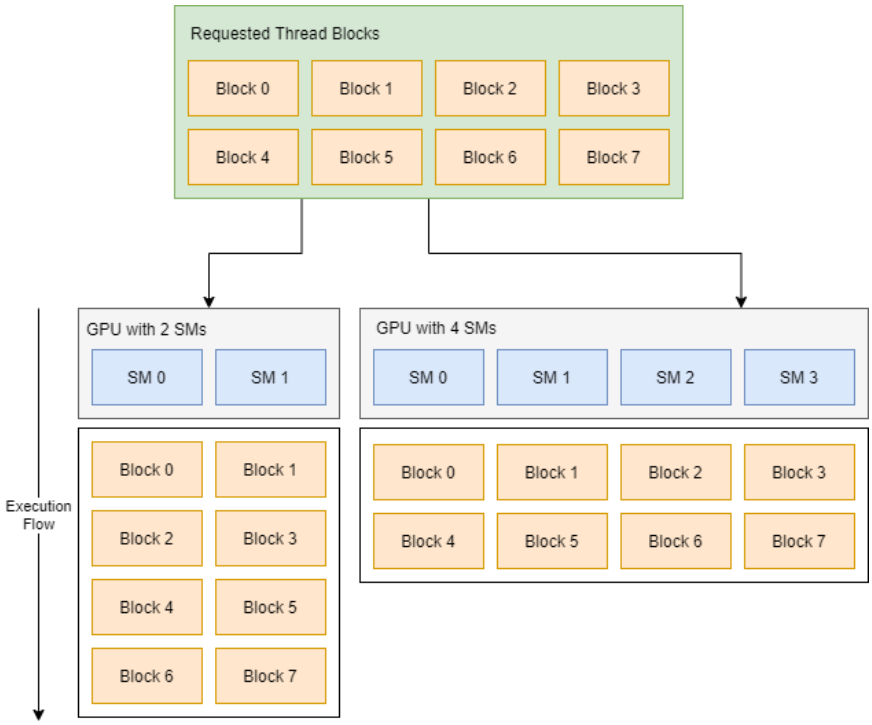


Figure 2.8: The GPU programming model of a CPU invoking a kernel with thread blocks that are then assigned to Streaming Multiprocessors (SMs). Redrawn from the CUDA documentation [Gui13].

2.7.1 The GPU Programming Model

While different API have differing terminology, the underlying GPGPU programming model remains similar. Here we refer to the terminology used by NVIDIA due to the techniques within this thesis utilising the CUDA API. GPGPU programming utilises the Single Instruction Multiple Data (SIMD) model, meaning that multiple threads run the same instruction on multiple data points simultaneously. To allow for the programming of GPGPU applications independent of hardware, the stream processing paradigm is used [BCL14]. This allows programmers to represent GPGPUs at differing levels of abstraction. The CPU invokes a kernel, consisting of a group of thread blocks that may be defined as 2 or 3 dimensional, containing a maximum of 1024 threads. The thread blocks are then assigned to a GPU core called a Streaming

Multiprocessor (SM) by a thread block scheduler in a load balanced fashion. These SMs contain their own L1 cache, read-only texture memory and shared memory. During execution, the SM processes these thread blocks in groups of 32 (or 64 in the case of AMD) called a warp, using the SIMD model.

Due to the unique hardware of GPGPU computing, specific design considerations must be made to effectively utilise the massively parallel nature of the GPU. Due to the SIMD model of warp execution, branching and divergence can cause significantly slower execution times. During flow control instructions (such as if, for and while) threads within the same warp may diverge, following different execution paths. These differing execution paths must be completed sequentially, increasing the number of instructions executed for that warp. Due to this, specific care must be taken to minimise the amount of differing execution paths within the same warp.

With the nature of GPU memory access, care should be taken when attempting to copy to or from memory. Memory is often the most important area for optimisation [Gui13]. Due to the peak bandwidth between device memory on the GPU being significantly higher than the bandwidth between host (CPU) and device (GPU) memory care should be taken as to where and when memory transfers occur. Significant performance increases can occur when intermediate results are calculated on the GPU. Even if no performance gains are shown during the processing of the intermediate data, avoiding unnecessary memory transfers between the host and device saves significant amount of time. Care should also be taken when accessing global memory. The shared memory that is allocated per thread block (48KB) has a much lower latency than global memory (>1GB). Effective use of shared memory to avoid global memory access can result in significant speedup [KWZ19, Gui13].

When access to global memory is required, it is best for threads to access consecutive memory locations. This allows for hardware optimisations to occur, with

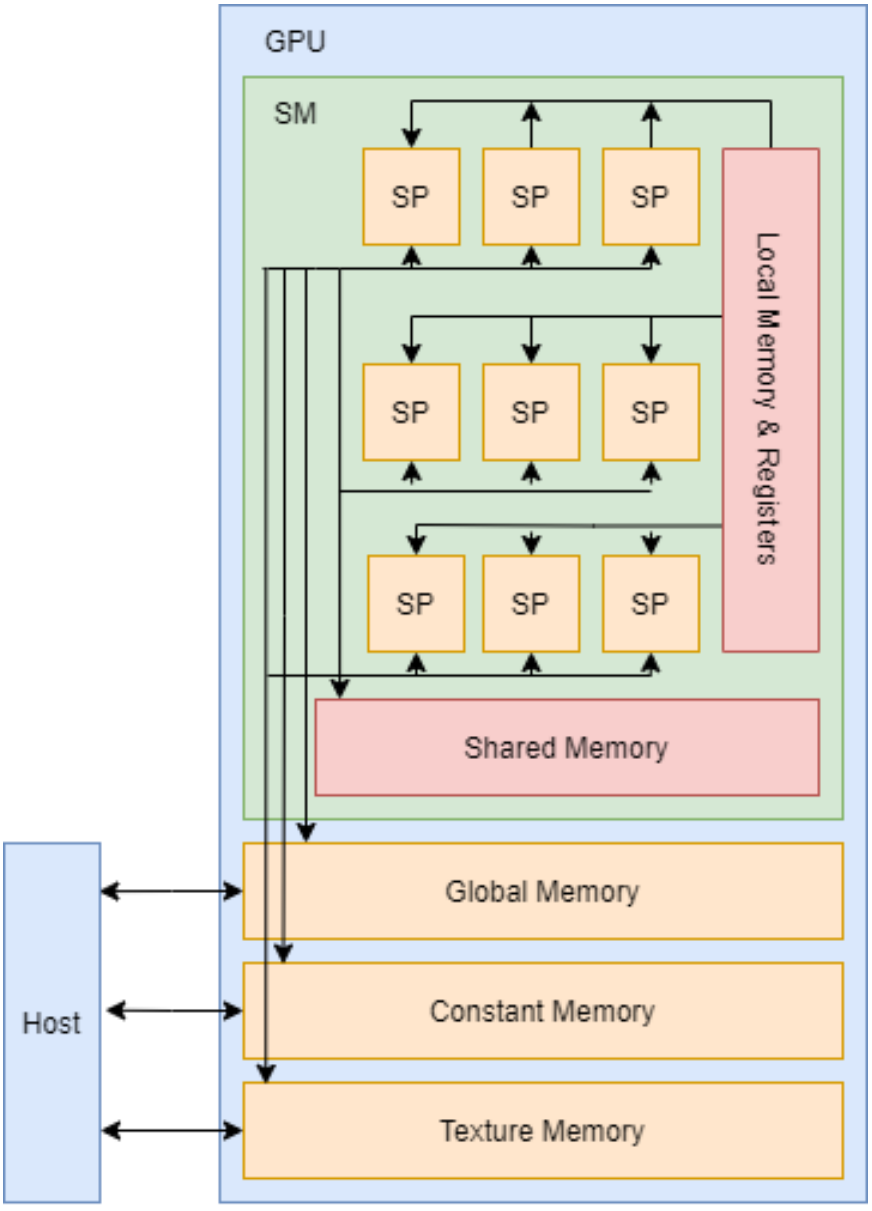


Figure 2.9: Abstraction of the main components of a NVIDIA GPU. Redrawn from [vO11]. Here we see a Stream multiprocessor (SM), and its Scalar Processors (SP), with the interaction between different GPU memory locations and the host (CPU).

consecutive memory access being combined into a single request, and thus reducing the number of memory operation that occur. This is known as memory coalescing. The amount of shared memory used by a warp can have a significant effect on the execution time of a kernel. If too much shared memory is used it could limit the number of active threads. The number of active threads against the theoretical maximum number of threads is known as the occupancy, with high occupancy typically indicating better execution times.

When designing GPU algorithms, care should be taken to optimise the algorithm to best utilise the hardware: A high number of active threads should be achieved at all times to maximise occupancy, warp divergence should be avoided through the use of fine graded data parallelism, shared memory should be used to minimise the use of the slower global memory and any reads or writes to global memory should be coalesced whenever possible. These algorithms are typically split into the use of a selection of parallel primitives. These include gather, scatter, scan and split [LLX19]. These parallel primitives are the building blocks of most parallel algorithms, allowing for typically CPU bound techniques to be reimplemented to utilise the massively parallel nature of the GPU.

While these considerations should be taken into account when writing performance critical GPU applications, recent libraries have been released that are designed for programmer productivity. with highly optimised common parallel algorithms and primitives implemented. One example of this is the Thrust C++ template library [BH12]. Use of libraries like these have allowed for the use of GPGPU computing in new areas of research without the need for specialist knowledge of GPGPU programming environment [OA18, WK13]. Due to this, GPGPU computing is now being used in a wide variety of disciplines, both using these programmer productivity focused libraries and custom written kernels. These include ecological applications [OA18],

computational biology, bioinformatics [NCTB17] and geospatial analysis.

2.8 Discussion and Research Gaps

While cartography and visualisation of movement data has a long and detailed literature, as can be seen from Section 2.3, the rise of big data and the larger datasets we are now able to collect effect all fields of research. Previous visualisation methods that rely on sequential processing, such as moveVis and DynamoVis [XD15, SRSW20], are now struggling to render the entirety of these datasets in a timely manner. It has been shown that real time interaction between multiple visualisation methods assists in understanding and investigation into datasets [ARH+15, SWS+19, WSN19]. It is here that GPGPU acceleration can be applied to allow for the processing of the millions of datapoints that are standard within toady's movement and environmental datasets, enabling real time interaction. The real time visualisation of both movement and ecological data could provide insights into the links between movement decisions of birds and the energetic costs of flying under different environmental conditions.

The benefits of real time interaction not only apply to visualisation, but to analysis as well. While there are large amounts of R packages and methods that allow for insights into the movement of a species, a large amount of these are sequentially processed, despite the fact that many of these methods lend themselves well to parallel processing. As datasets continue to grow these analysis methods will only require longer computation time, furthering the disconnect between any changes to parameters and results. While some methods are now beginning to be aware of the need for scalable solutions [PBS+18, HLZL20], the scalability of many methods (Section 2.6) can be improved. It is here that GPGPU computing can allow for the real time parallel processing of millions of datapoints in real time.

Finally, as we begin to visualise more data, care needs to be taken to avoid issues

in the readability of the visualisations we create. This is an issue especially prevalent with space time cubes and trajectory data [AA07]. Aggregation has been shown to allow for insight into the space utilisation of movement with success using networked clusters to process billions of datapoints of maritime data [GWD20]. While this allows for large datasets to be processed, there are significant hardware limitations processing data with networked clusters. We believe that there is a middle ground that can be achieved with far more available hardware through the use of GPGPU computing.

Chapter 3

GPU Accelerated Globe Rendering and Environmental Dataset Switching

3.1 Introduction

Environmental context is an important factor in the investigation of movement data, with resource availability significantly effecting the movement of species [DV10b]. Global datasets that describe these external environmental factors, such as land cover, elevation, temperature and vegetation, are now available at high spatial and temporal resolutions [Did15, ZW15, MS19] but with the sizes of these datasets there is a struggle to visualise the quantity of data points involved while also supporting visualisation in real-time. As of writing there are few available packages to visualise and investigate movement data interactively containing millions of data points on a single workstation. Pre-rendered visualisations are available, but these have limitations. For example, Schwalb-Willmann et al. [SRSW20] states that "rendering an animation of 500 frames with 25 frames per second, resulting in an animation duration of 20 s, takes approximately 25–35 min" for their R package moveVis. These visualisations, while animated, restrict the user's view to a predetermined point of view

without re-rendering, requiring repeated long computation times. This leaves a disconnect between the adjustment and resulting visualisation, in addition to extending the amount of time required to investigate differing processes.

This is especially prevalent for the visualisation and investigation of large scale movement data, such as migratory datasets. As a result of the distances involved within these datasets, it is difficult to dynamically visualise and investigate the effect that these environmental factors have on the movement of a tagged species over a large area in real-time. The ability to interact with and investigate these datasets in combination with global environmental data would allow for researchers to gain insight into the decisions being made by the tagged species during a migration through data driven approaches [WTB+20]. While most systems have enough memory to accommodate these datasets to a suitable resolution, CPUs struggle with the computation required for real-time visualisation of millions of data points [SRSW20].

The techniques presented here achieve real-time visualisations via the use of GPUs and the CUDA programming environment, processing the millions of points within these global datasets at interactive speeds. Via the use of the CUDA parallel programming platform it is possible to visualise a single timestamp of the MODIS/Terra Vegetation Indices dataset (25,920,000 data points) [Did15] in only 5 milliseconds with a NVIDIA GTX 980, allowing for rapid switching between datasets, or between timestamps of the same dataset, to visualise the global changes in the desired datasets overtime. This, in combination with tracking data of large scale migratory species, enables the visualisation of the effect that environmental factors have on migratory movement in real-time, rather than a pre-rendered video visualisation. This allows for researchers to harness the power of their GPUs to allow for the exploration of multiple studies on a global scale and provide insight into the movement of species with environmental context.

3.2 Methods

As discussed in Chapter 2.2 when visualising global environmental data, many online mapping methods still use the Mercator projection [BFUY14], despite its issues [Jen12]. With differing users preferring different projections for global visualisations [SJWS15] care was needed in selecting the method of visualisation. To avoid the reprojection of data, while still minimising distortion, a 3D globe visualisation is used. The use of a 3D sphere has been used to more clearly represent the globe with less distortion compared to a 2D projection [LHW20, GKD10]. This essentially creates an orthographic projection, with the projection being recalculated in real time allowing for the camera to be moved such that the area of minimal distortion (the focal point) can be moved as the user desires.

To create the geometry a CUDA kernel is invoked, creating the ECEF sphere at a given resolution, allowing for adjustment. The visualisation of height data is also possible, however due to the exceptionally high resolution of today's digital elevation models (the STRM DEM has a resolution of 1 arc second [FRC+07]) a lower resolution is used, with a smaller focus area being rendered at a higher resolution. The digital elevation model is used to calculate the surface normal of the globe by first calculating the normal as if the DEM was mapped to a flat plane, and then wrapping it around the unit sphere. If the DEM is not desired the nature of a unit sphere allows for the position to also be used as the surface normal, saving memory. While methods for level of detail (LOD) rendering exist to allow for smoother and more efficient rendering [BAB+17] it is not the focus of this research. Due to the differences in scale between the diameter of the globe and the height of the terrain making visualisation of height values difficult to see, an adjustable height multiplier is used to exaggerate terrain features, with a CUDA kernel updating the geometry as needed when the height multiplier is changed by the user. With the camera able to rotate and zoom

to different parts of the globe this underlying geometry can then be used to visualise both environmental data, digital elevation data and movement data.

3.2.1 Environmental Dataset Loading

Due to the significant increase in data availability from many different sources, a large number of geospatial formats (such as HDF4, HDF5, NetCDF etc.) for vector data now exist. These data formats vary massively, with many of these incompatible. To allow for global datasets from multiple data sources to be visualised together, without the need for specialised knowledge to convert formats, the Geospatial Data Abstraction Library (GDAL) is used [War08]. GDAL is a translator library commonly used in the GIS community that allows for the reading and modification of raster data from many of the most popular geospatial data formats. More detail on environmental dataset loading is found within Chapter 6.

Upon selecting a geospatial file containing the dataset to be loaded, for example the MODIS 0.5-degree temperature dataset, all metadata is loaded and shown to the user. This metadata contains the data type of the dataset, 16-bit integer in the case of the MODIS temperature dataset, the size of the dataset and any other associated layers. A prompt is then given to the user to input the metadata needed for the visualisation. First is the timestamp of the dataset, with a temporal resolution if the dataset layers describe differing timestamps. Next is the desired name of the dataset, if one cannot be detected within the file, and finally the registration of the dataset, either pixel or point. The user then selects the method of interpolation desired for both spatial and temporal dimensions, with bilinear and nearest neighbour available for the spatial dimensions. To allow for the visualisation of averaged data over a specific time (such as the MODIS monthly NDVI [Did15]) the previous timestamp can be selected if desired, with closest and linear interpolation also available for the

temporal dimension.

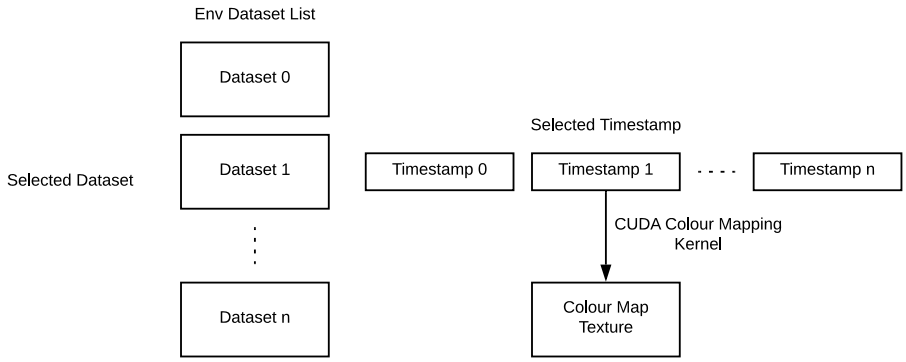


Figure 3.1: The environmental dataset list data structure. The environmental dataset list contains pointers to a set of datasets with assigned timestamps under a common name. This allows for fast access of the requested environmental dataset. These datasets can then be sent to the GPU for processing.

With this information, a list can is used to store each separate environmental dataset, which in turn contains a vector storing the rasters for each timestamp in chronological order to allow for faster access. As time advances within the visualisation this timestamp list can be queried to find the desired datapoints. The environmental dataset class makes use of C++17 std::variant to allow for datasets of differing types to be within a single vector. This is combined with C++ templating to allow for multiple different data types to be loaded, either as separate datasets or as timestamps within the same dataset. This allows for the precision of the dataset to remain unchanged during visualisation. By default, upon initially loading the geospatial file, the data is not loaded onto system memory until specifically requested by the visualisation, where it then remains. This is to allow for the loading of environmental datasets that have a significant number of datapoints, too many to be stored onto system memory. Single points can then still be queried without loading all of the data (This is more important for annotation of movement data, covered in Chapter 4).

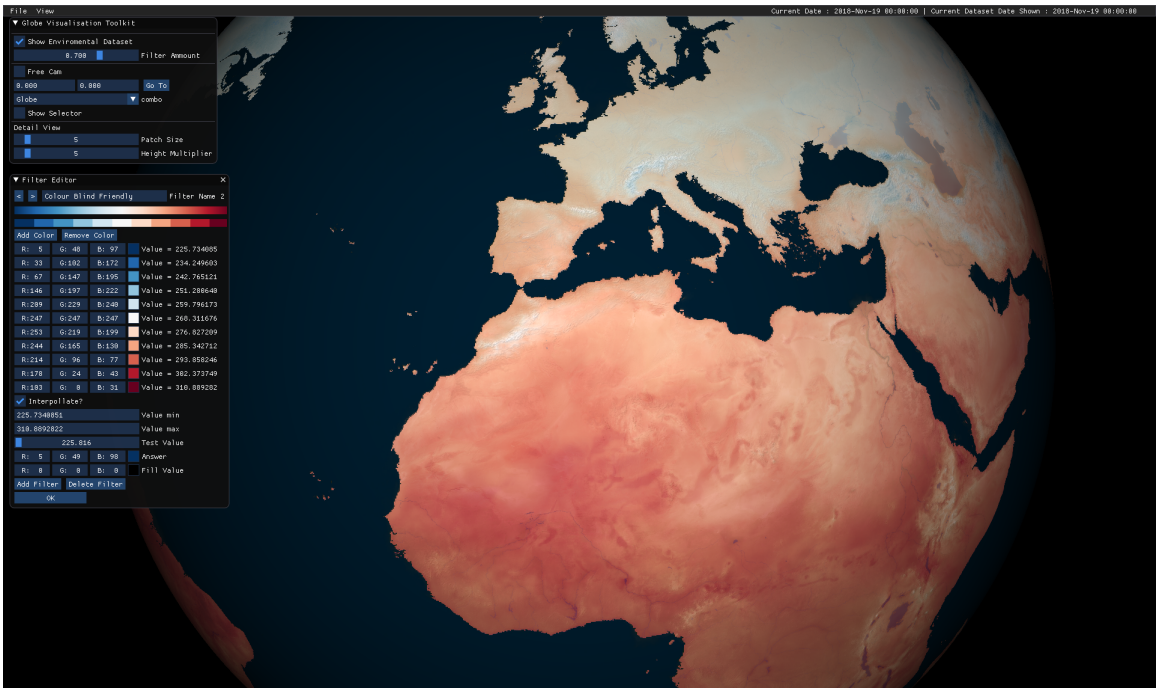


Figure 3.2: Here we show the the ERA5 Temperature dataset [MS19], with a temporal resolution of 6,483,600 datapoints per hour. The visualisation benefits greatly from GPGPU acceleration, with less than 5ms required for switching between timestamps. Due to the high temporal resolution dataset switching occurs frequently, but GPGPU acceleration allows for a global visualisation of the environmental variables alongside trajectory data.

3.2.2 GPU Accelerated Dataset Switching

Before creating the visualisation, a user selects the colour mapping desired for the current dataset. This uses a modified version of the Spectrum C++ colour mapping library by Roberts et al [RMAL18], enabling the use of a large number of different colour mapping methods, such as diverging, sequential and qualitative colour maps, with optional interpolation between colours. Before any visualisation can be created, a colour mapping is needed. Upon loading the toolkit for the first time, a basic rainbow colour map is provided. Users can then either adjust this colour map to suit their needs or create an entirely different colour mapping. Any colour maps created this way are saved to an external JSON file that is loaded the next time the user opens

the toolkit. This allows for researchers to create a number of different visualisations, either for different environmental datasets, or to highlight specific areas of the same dataset. This JSON file can then be shared with researchers who also have this toolkit to visualise the same data in the same way, aiding in collaboration.

During visualisation, if time advances past the next timestamp or the user manually changes the current time, an update request is sent to the CUDA controller class. This also occurs if the colour map of the visualisation is changed. Upon receiving the update request, the colour mapping is then sent, alongside the appropriate environmental dataset metadata, to the CUDA controller class for processing. The CUDA controller class handles all of the heterogeneous communication between the sequential processing of the CPU and the high throughput mass parallel GPU.

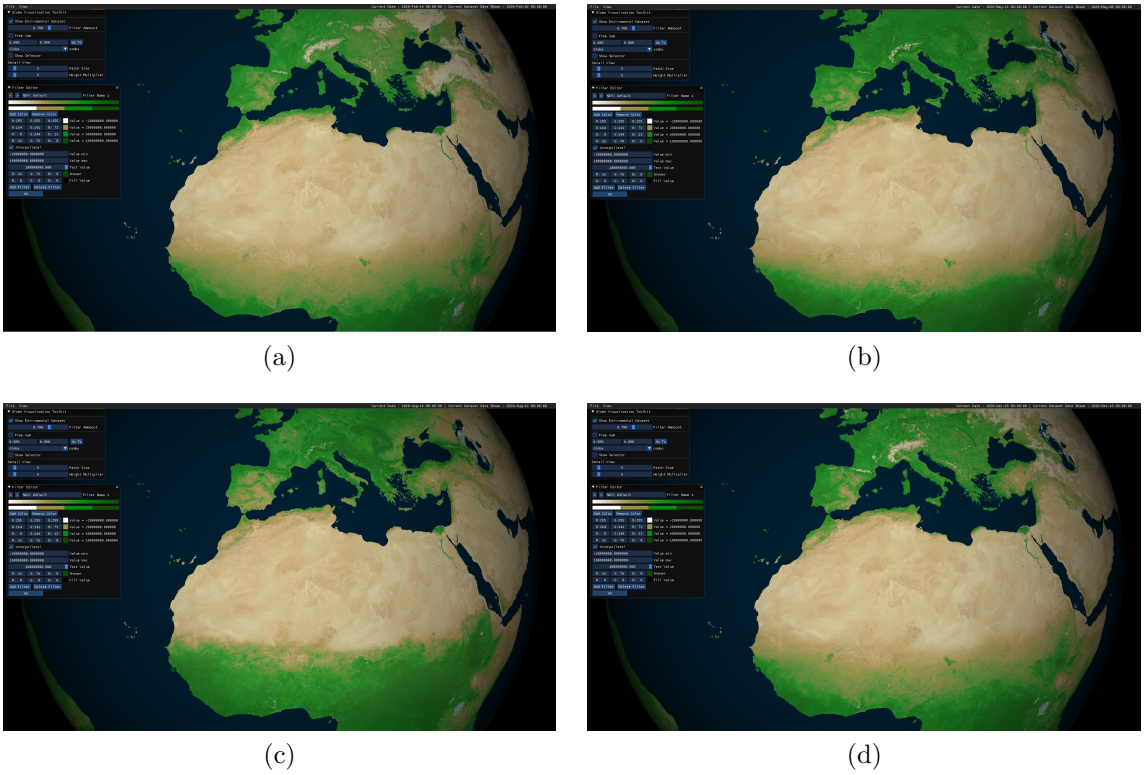
To facilitate the real time rapid switching of global datasets GPGPU acceleration was applied. The OpenGL CUDA interoperability is used to allow for the result of a CUDA kernel to be applied directly to a texture. This avoids using a VBO to store the environmental data and using the fragment shader to interpolate a colour mapping across the globe every frame (the pitfalls of which are discussed in Section 3.3) or using existing plot packages within R that rely solely on the CPU [Wic16, SRSW20].

Due to the time intensive nature of memory transfer when compared to processing in large datasets, memory optimisations offer the largest improvement in performance. To ensure the fastest possible transfer rate between the host and device memory pinned memory allocation is used for the environmental dataset list. While the allocation of pinned memory, via the use of cudaMallochost, is slower than the standard allocation of memory using malloc, the GPU can directly access this pinned, also known as page-locked, memory. This added allocation time will only affect the loading of environmental datasets, not the time sensitive dataset switching of the visualisation. With direct access to the memory, the GPU can transfer the

environmental dataset with minimal assistance from the CPU.

To further increase the speed of processing concurrent memory transfer and execution is utilised, achieved via the use of the CUDA streams [Har12]. Due to the use of pinned memory, allowing the GPU direct access to the data, this is possible. First the data is split into equal parts that will be processed by different streams. Each stream will then initialise transfer of their section of the dataset to be processed, once the section of the dataset that the stream has been assigned has been fully transferred execution of the colour mapping kernel begins. Once complete the transfer to texture memory can be commenced. The hardware available does affect this process, however. NVIDIA GPUs contain engines utilised for various tasks, such as copy and execution engines. It is the use of these that allow for processing of data in different concurrent streams. It should be noted that in the comparison of breadth first and depth first execution it is usually assumed that all memory transfers and kernel execution require the same amount of time. This is not the case here, as the results of the CUDA kernel are copied to a texture, resulting in a device to device memory copy, rather than to host memory. This is far faster than the transfer of host to device. Here, depth first execution is used. It should also be noted that due to the nature of transferring the results of the kernel to a texture rather than back to host memory, there was little difference in performance between the scheduling methods.

Finally, to minimise the number of memory operations performed, a memory buffer is created on application start that waits to receive the next set of environmental data. This memory buffer is adjusted if the resolution, and thus the size, of the environmental data is larger than the buffer. The buffer is initialised to a small size and when a dataset with a higher resolution is added the buffer is adjusted and then remains at this larger size. This ensures that the smallest amount of memory needed is allocated, while still avoiding the continuous reallocation of memory.



(a) (b)

(c) (d)

Figure 3.3: The MODIS/Terra Vegetation Indices dataset, containing 25,920,000 data points per timestamp. Here four months are visualised: Feb (a), May (b), Aug (c) and Dec (d).

3.3 Results

To assess the performance of GPU acceleration, three methods of loading and then rendering the data are compared. The first method utilises the geometry CUDA kernel to update the underlying geometry of the globe to match the resolution of the desired dataset. This is done by first updating the size of the VBO that stores the geometry of the globe so that there is enough memory and then calling the CUDA kernel to calculate the positions on the unit sphere. The environmental data can then be sent to the VBO, either as RGB values with the colour mapping already applied by the CPU (this is done as a baseline to assess the speedup of GPU acceleration) or as raw values which can then be mapped to a colour within the fragment shader

Method	Idle Framerate (fps)	Dataset Switch (ms)	Active Framerate (fps)
CPU	2254	184.4	5.43
SSBO	95	29.87	38.62
CUDA	2856	5.354	186.7

Table 3.1: Table comparing the idle (framerate of the visualisation with no current environmental dataset switching) and active (framerate when the visualisation is switching environmental dataset at the start of every frame) framerates of the three visualisation methods, CPU only, SSBO with colour mapping fragment shader and CUDA optimised colour mapping. The environmental dataset used for testing was the MODIS/Terra Vegetation Indices dataset, containing 25,920,000 data points.

using the Spectrum library rewritten in GLSL. The an SSBO is uesd in an attempt to avoid the large VBOs needed to store the vertex data. Here, the SSBO is instead updated to allow for sufficient memory to contain the environmental dataset, which is then sent to the newly allocated SSBO. The fragment shader then applies the colour map. Due to the SSBO not being tied to vertex data the resolution of the geometry can be far lower, with a significant amount of memory saved for larger datasets. The final method uses the CUDA accelerated method with the OpenGL CUDA interoperability to dynamically update a texture. The performance increase of the memory optimisations are then compared. Finally, the impact of using a float datatype and a short int (int16) on differing dataset sizes is compared. It should be noted that the results for the CPU based methods do not make used of multi threading for both this chapter, and subsequent chapters. This is due to the fact that many R package implementations, both in R and C++ code called from R, do not make use multi threading.

As can be seen from Table 3.1 significant performance increases can be seen from the application of CUDA. The idle framerate is defined as the framerate of the visualisation when no dataset switch is occurring, with the active framerate measured when the visualisation is continuously switching environmental dataset (this arises

when the playback rate of the visualisation is high, or when the temporal resolution of the environmental is very high). The environmental dataset used for testing was the MODIS/Terra Vegetation Indices dataset, containing 25,920,000 data points. While the CPU method has similar idle framerate as the CUDA optimised method, the active framerate is much lower, owing to the expected increased computational time of calculating the colour map on the CPU. The SSBO method increases performance for dataset switching, reducing to 29.87ms, this is turn leads to a higher active framerate of 38.6fps, allowing for the interactive investigation of these datasets. It should be noted however, that due to the increased computation needed within the fragment shader the idle framerate drops significantly. This is due to the recalculation of the colour map every frame, even if the environmental dataset remains the same. While this still allows for interactivity, the idle framerate will continue to reduce as larger environmental datasets are used. The CUDA method allows for both idle and active framerates to remain high, with the CUDA optimisations reducing the dataset switch even further to 5.35ms, and maintaining a high idle framerate exceeding 2000fps.

Table 3.2 and Figure 3.4 show the effect of each optimisation step on the overall computation time for one timestamp of an environmental dataset, and compared these to the initial CPU implementation. Here, four different GPUs were used for testing, the RTX 2080, Titan Xp, GTX 1080 and GTX 980. The initial CUDA kernel used showed significant speedup alone, with all four cards able to achieve a visualisation in one eighth of the time of the CPU equivalent, while still maintaining a high idle framerate. Optimising the memory transfer via the use of pinned memory and concurrent copy and execution allowed for further performance increases, with the RTX 2080 creating the visualisation in 8.71ms. Finally, by managing the memory buffer that receives the environmental dataset for processing, a further significant

GPU Tested	GPU Optimisation Step	Time (ms)	Step Speedup	Total Speedup
CPU Only	Sequential CPU Initial CPU Benchmark	184.44		
RTX 2080	Kernel 1 Sequential transfer from host memory	14.37	12.84x	12.84x
	Kernel 2 Sequential transfer from Pinned memory	10.92	1.32x	16.89x
	Kernel 3 Overlapping Kernel Execution and Data Transfers	8.71	1.25x	21.18x
	Kernel 4 Pre-allocation of GPU buffer	4.51	1.93x	40.90x
Titan Xp	Kernel 1 Sequential transfer from host memory	15.88	11.61x	11.61x
	Kernel 2 Sequential transfer from Pinned memory	12.23	1.30x	15.08x
	Kernel 3 Overlapping Kernel Execution and Data Transfers	10.46	1.17x	17.63x
	Kernel 4 Pre-allocation of GPU buffer	4.72	2.22x	39.08x
GTX 1080	Kernel 1 Sequential transfer from host memory	17.01	10.84x	10.84x
	Kernel 2 Sequential transfer from Pinned memory	12.78	1.33x	14.43x
	Kernel 3 Overlapping Kernel Execution and Data Transfers	10.01	1.28x	18.42x
	Kernel 4 Pre-allocation of GPU buffer	4.87	2.06x	37.87x
GTX 980	Kernel 1 Sequential transfer from host memory	22.62	8.15x	8.15x
	Kernel 2 Sequential transfer from Pinned memory	19.91	1.14x	9.26x
	Kernel 3 Overlapping Kernel Execution and Data Transfers	15.52	1.28x	11.88x
	Kernel 4 Pre-allocation of GPU buffer	5.05	3.07x	36.52x

Table 3.2: Speedup of colour mapping kernel compared to initial sequential CPU implementation using the NVIDIA RTX 2080, Titan Xp, GTX 1080 and GTX 980 GPUs.

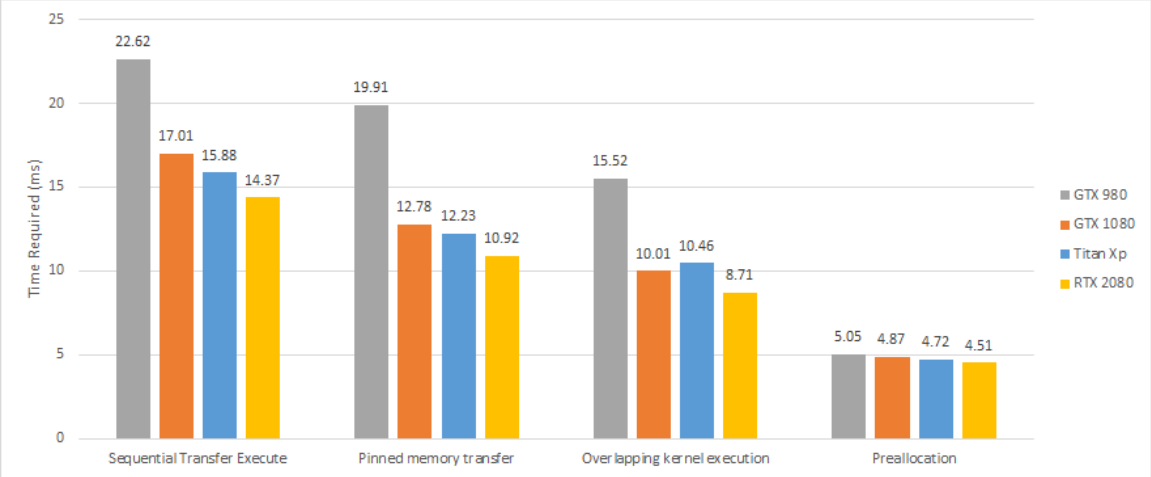


Figure 3.4: Final results of the optimisation steps on the differing GPUs. With the higher memory and clock speeds the RTX 2080 overtakes the other cards in earlier optimisation results. With the full optimisation however, the lower end GPUs, such as the GTX 980, are able to process one timestamp of the NASA MODIS vegetation dataset (just below 26 million datapoints) in a similar time. With all optimisations applied, all cards can achieve a real-time visualisation.

speedup is achieved, with all four GPUs able to create the visualisation in approximately 5ms. It is here that the benefits of the optimisations used are highlighted, even with lower performance GPUs, large amounts of environmental data can still be visualised in real time. Adjusting the block and grid size had little effect on the overall calculation time of the colour map, with the block size adjusted from 32 to 1024. A very minor speedup of 0.3ms was observed at a block size of 1024.

Finally, Table 3.3 shows the computation time required for global environmental datasets at varying sizes. These environmental datasets were generated by resampling the MODIS/Terra Vegetation Indices dataset to higher resolutions. Here we can see that even with a global dataset that contains over 414 million data points, approximately a 1km resolution at the equator, a usable visualisation can be observed. The method of storing this dataset can be seen to have an effect here, with datasets that have been compressed to have all datapoints represented by a 16 bit integer requiring less time than the larger, thus more computationally expensive, 32 bit floating point

Resolution (deg)	N	Time (ms)		FPS	
		int16	float	int16	float
0.5	259200	0.13	0.23	7567	4351
0.1	6480000	1.46	2.23	680	447
0.05	25920000	5.35	8.81	186	113
0.025	103680000	20.54	34.89	48	28
0.0125	414720000	83.08	138.29	12	7.2

Table 3.3: The time required to switch environmental dataset, calculate the colour map and render at varying resolutions and data types of global data using the NVIDIA Titan Xp.

number. It should be noted that even with these exceptionally large datasets, the idle framerate was not significantly affected.

3.4 Conclusion

Environmental context is an important factor to consider during the investigation into movement of animal species [DBC+15]. Visualisations of global environmental datasets allow for researchers to investigate the effect of the environmental context for species that cover large sections of the globe, such as migratory studies. These studies, however, struggle to visualise the vast amount of data involved in real time. The ability to interact with a dataset in real time has been shown to improve comprehension [PM15], and allowing researchers to visualise their movement data in an interactive environment with these larger environmental datasets could allow for researchers to gain insight into the movement of tagged species.

Here we present a GPGPU accelerated visualisation method that allows for researchers to visualise the large global environmental datasets used to investigate the movement of migratory animals. Researchers can load multiple environmental datasets, each with multiple timestamps, and switch between them freely, either to a different timestamp as the visualisation progresses, or to an entirely different dataset.

Users can apply a multitude of differing colour maps, which can be applied to any number of environmental datasets and save them to an external file to allow for collaboration between researchers. Through the use of CUDA optimisations, such as pinned memory, concurrent copy and execute and buffer management, the toolkit is able to create a visualisation of the MODIS/Terra Vegetation Indices dataset that contains over 25 million datapoints in under 5ms. This visualisation is not restricted to high spec GPUs, with lower end GPU able to perform with similar results. This is in significant contrast to current visualisation packages that require 25-35 min to create a comparable visualisation [SRSW20]. It is hoped that researchers will begin to use this interactive visualisation method to investigate the effect of differing environmental variables have on the movement of tagged species.

Chapter 4

GPU Accelerated Resampling, Analysis and Event Detection for Migratory Datasets

4.1 Introduction

As the amount of data we collect increases, the methods that we use need to evolve to handle the larger dataset sizes of today. It is no longer uncommon to have millions of datapoints for a single tracked animal within a study, with a study made up of a multitude of different animals. This is especially the case for migratory studies, where the distances travelled and their annual nature creates datasets that span large distances over long periods. For example, the eastern flyway spring migration of adult white storks dataset from Rotics et al. [RKT+18] contains 1.2 million relocations for only 36 tracked animals. This data not only contains positional data, but also the data from other sensors, such as accelerometers used to estimate energy expenditure. As we begin to investigate the world around us, and the effect we have on both the environment and the differing species within, these datasets will only continue to grow. It is thereby crucial that researchers from differing disciplines collaborate to ensure that the toolsets we use are able to process these larger datasets effectively and accurately.

Recent methods have begun to use the power of distributed computing and networked clusters to process these datasets [Mro18, GWD20]. While effective, these methods do not lend themselves to single workstations. As geographic information systems continue to evolve, and with open source solutions becoming more popular, these methods, while effective, are limited by the complexities of working within a distributed system environment. Here, we propose moving away, at least partially, from processing datasets using distributed systems. While they are proven to be effective in processing even billions of datapoints in a timely manner, the development of toolsets and methods that allow for a single workstation to process data is a benefit to both those who have access to the networks clusters and those who do not. The recent rise of GPGPU computing is now allowing researchers to analyse their data using a single workstation faster than ever before, while still allowing for these methods to be scalable for networks. GPGPU accelerated analysis also allows for the analysis of datasets that may be too small for distributed computing to be efficient but are larger than the conventional CPU to handle in a timely manner. The limits of CPU based methods become particularly apparent when the analysis being performed has a number of different parameters. As researchers attempt to explore their data via data driven methods [WTB$^+$20], the amount of computation time can significantly increase, as even with domain specific knowledge, parameters may need adjustment multiple times to achieve the intended result. Waiting for results of similar analysis, especially for the same analysis with different parameters, can cause a disconnect between the parameter adjustment and its effect on the results.

Here we present a number of methods that are commonly used in the field of movement ecology, accelerated with GPGPU computing to allow for interactive calculation of both first and second order movement metrics. We also perform more involved methods, such as the resampling of data to a common time interval, the

detection of migratory periods, turning point detection and movement metric thresholding to detect 'event points', relocations within a trajectory that may provide some insight into the movement of tagged species. While GPGPU programming may be intimidating to those without a computer graphics background, we present these methods as a collection of parallel algorithms and primitives. This provides a level of abstraction, allowing for researchers to think in terms of manipulating the underlying data, rather than the complexities of GPGPU computing. We then compare the computation time required for CPU based methods with their GPGPU counterparts. We allow for users to perform these metric calculations in real-time, visualising the results alongside the underlying trajectory, made possible by a heterogeneous computing environment. This coupled with the environmental context visualised from Chapter 3, allows researchers to glean insights into the movement of tagged species, and then consider these movement metrics with environmental context in real-time.

4.2 Parallel Primitives and Algorithms

In this chapter, we make extensive use of commonly used parallel algorithms and primitives to enable GPGPU acceleration of methods used in movement ecology. Literature shows that parallel primitives have been used as the building blocks for more complex algorithms, from eco-evolutionary simulations [OA18] to the calculation of NDVI [ACHLRZ14]. Due to the methods within this thesis using the CUDA programming environment, the thrust library is used. Modelled after the C++ standard template library, thrust is a template library for CUDA that emphasises programmer productivity via the use of pre-optimised parallel algorithms. The boost compute library does allow for users without an NVIDIA card, due to being developed with OpenCL [Szu16] and has many of the same parallel functions available. To allow for others to reproduce the results, we present all analyses performed as a set of parallel

Parallel Function	Description	Relevant Papers
Transform (Elementwise operation, Embarrassingly parallel)	The simplest of parallel primitives. Takes one or more inputs of the same length and outputs an array of equal length according to the function given.	[Gui13]
Reduction	Combine each element within the array to a single value. For example, the sum of all elements within the array, or the minimum and maximum elements of an array.	[San02, RT04]
Scan (inclusive scan, exclusive scan, prefix sum)	For each value within an array, calculate the sum of all preceding elements (including the current value in the case of inclusive scan). This is also known as a prefix sum. This operation does not have to be a sum however.	[SHZO07, SHG+08]
Partial Reduction (reduce by key)	Similar to reduction, instead of reducing to a single value, instead use an array of equal length containing a key. The resulting array contains separate reductions of values that had the same key.	[HGP+18]
Stream Compaction (copy if)	Remove all values from an array that do not pass a defined predicate. Often used to retrieve sparse data from large datasets.	[BOA09, BM13]
Merge	Merge two arrays, such that the length of the resultant sorted array is the sum of the length of input arrays.	[GMB12, BM13]

Table 4.1: Table containing the parallel algorithms used within this chapter.

primitives and algorithms commonly included within these libraries. Table 4.1 shows each of the parallel functions used with a brief description and papers that contain further detail on the algorithms used.

4.3 Metric Calculation

A number of different metrics that describe the movement of a tagged species are currently implemented. While the calculation of these metrics would usually require the trajectory to be sent to the GPU for processing, we already visualise the trajectories by transforming the points to whichever visualisation method is preferred. This means that the trajectories are already stored within device memory, available via the OpenGL interoperability. This allows for the contents of the VBO to be used directly within a kernel, with only the timestamps to be sent to the GPU if required. The results of these metrics can then be immediately placed within a VBO for visualisation. The currently implemented metrics are shown in Table 4.2.

Metric	Description	Relevant Papers
Daily Distance	The total distance travelled by an individual each day. Returns one value for each day.	[SAF+20]
Daily Net Distance	The distance between the first and last relocations of each day. Used for migration detection.	[SAF+20]
Step Distance	A first order metric where the distance between each relocation is calculated using the haversine formulae.	[DWL08, SAF+20]
Time Step	The time duration between each successive relocation.	[DWL08]
Step Speed	The assumed velocity of the tagged species assuming constant speed between each relocation. Calculated using step speed and time.	[CDRC09, DWL08]
Net Squared Displacement	The distance between the first and each subsequent relocations. Used to estimate diffusion behavior and migration patterns.	[SAE16, SAF+20, BRPY+15b]
First Passage Time	The time required for crossing an endpoint defined by a circle around each relocation and a given radius, both forwards and backwards in time.	[MLM09]
Turning Angle	The relative turning angles between each relocation.	[Bat81, ME02, PBS+18]
Pseudo-Azimuth	The Azimuth value between each relocation that is clamped between the values of 0 and 360. Used to identify parallel movement such as migratory corridors.	[LGWK13]

Table 4.2: The movement metrics currently implemented within the Global Animal Movement Toolkit (GAMT). More details on GAMT in Chapter 6.

4.3.1 Transforms

The first of the metrics covered here are known as being 'embarrassingly parallel', where the the calculation lends itself exceptionally well to being processed in parallel, with little or no communication between threads necessary [HSLS20]. These metrics are a common occurrence in movement ecology and lend themselves well to the throughput nature of GPGPU computing. These initial transformations are usually used as an input for other methods. For example, net square displacement (NSD) has been used to classify migratory behaviour using latent state models [BRPY$^+$16]. NSD is obtained by calculating the straight line distance between the initial point of a trajectory and each relocation. This is done by placing the starting point within constant memory, allowing for fast access to each thread, and each thread calculating the distance between its given relocation and the starting point within constant memory. Other metrics can be calculated using an element wise transformation as well, such as turning angles, step distance and time step. Here, each thread processes the assigned relocation by accessing the current relocation and the one preceding it. These metrics can then be used to calculate others. For example, the time step and step distance metric can then be used to calculate the step speed.

Turning angles of a trajectory are commonly used to describe the changes in the direction of movement [Bat81, ME02]. This stepwise metric is used in a large number of methods, such as correlated random walks [TOSW15] and detecting decision points [AA11, PBS$^+$18]. To calculate the turning angles between two points Vincenty's formulae are used. This transforms a vector containing the trajectory with size N to a vector of size $N1$ that contains the sub trajectories between each relocation, represented by the starting and final azimuths for each step within a trajectory. From here, turning angles can then be calculated in parallel by calculating the difference between the azimuths. The starting and end points are assumed to have a turning

angle of zero. The Pseudo-Azimuth of each point can also be calculated from this array.

The pseudo-azimuth metric has been used in the identification of parallel movement, most notably migratory corridors, either within the same trajectory or between different trajectories [LGWK13]. Instead of calculating the difference between successive azimuths, the pseudo-azimuth is calculated by adding 180 to the azimuth of the relocation, then multiplying this value by two. Then, if the value is above 360, subtracting 360 to clamp the value. This is performed to each relocation using a parallel transformation, with the final point assumed to have the same azimuth as the preceding point, thus returning a vector of size N. These values can then be used to identify migratory corridors.

Not all metrics have a one to one mapping, processing only the immediate relocation. For example, First Passage Time is a measurement of the length of time a trajectory remains within a given radius for each relocation and is often used as an indicator for foraging behaviour [MLM09, FT03, LCDC14]. This metric can be calculated by first performing a backwards pass, calculating the distance from the initial relocation and each relocation preceding it, stopping and noting the time difference when the distance threshold is passed. Once complete, the threads are all synced, and a forward pass is performed, again noting the time required to pass the threshold. The two are then added to obtain the FPT. While this may cause some issues for warp divergence due to each thread passing the threshold independently, each warp of 32 threads processes successive relocations within the trajectory. This means that unless there is a significant gap in the trajectory with large amounts of movement, or if there is the beginning of a stopover at the beginning or end of a warp, each thread will process a similar amount of data. The methods described here use a modified version of the adehabitatLT R package by Calenge C [Cal06] to calculate FPT on the

GPU in parallel.

It should be noted here that to avoid warp divergence, all distance calculations are done using the haversine distance measurement by default. This is due to Vincenty's formulae being iterative, converging to the solution. Each calculation of distance or turning angles, while more accurate, may require a variable number of iterations within a warp, causing warp divergence.

4.3.2 Reduction

The next metrics make use of the reduction and partial reduction parallel functions. A common use of reduction is the calculation of summary statistics, such as the mean, median, standard deviation, minimum and maximum of any desired movement metric. This is achieved by using a slightly modified summary statistics example from the thrust library [BH12] that uses the formulae from Chan et al. to combine arbitrary sets [CGL]. Partial reduction is also used to calculate metrics over specific predefined periods. For example, the distance travelled over a day is calculated by first transforming the relocations into step distances, which are then partially reduced using the timestamps as a key.

This can be modified to allow for the net distance between the first and last relocations of the day to be calculated. Instead of using a sum reduction on the step distances using the daily timestamp as a key, the positions of each relocation are first transformed into a struct that contains the earliest and latest timestamps. As the points are reduced, the minimum and maximum timestamps and their current positions can then be updated. With the partial reduction complete, the output contains the first and last timestamps that occur within the same day. These are then transformed to distances using the haversine or Vincenty's formulae.

For spatial analysis, reduction can also be used for the calculation of the centre

point of a set of relocations. This is calculated by first transforming each point to ECEF geocentric coordinates using Equations 4.3.1 - 4.3.3. Here, the geodetic point $P(\lambda, \phi)$ has a longitude λ, latitude ϕ, and is transformed into its geocentric point $p(x, y, z)$. We then perform a sum reduction, this sum is then used to obtain the average point of all geocentric points $p(\bar{x}, \bar{y}, \bar{z})$ (Equation 4.3.4). While this point will most likely not land on the ECEF sphere, during reprojection back to geodetic coordinates we can assume that there is an altitude of 0 to obtain the centre point in geodetic coordinates. This is shown within Equations 4.3.5 and 4.3.6. An example showing each variable can be found in figure 2.3.

$$x = \cos(\phi)\cos(\lambda), \tag{4.3.1}$$

$$y = \cos(\phi)\sin(\lambda), \tag{4.3.2}$$

$$z = \sin(\phi), \tag{4.3.3}$$

$$p(\bar{x}, \bar{y}, \bar{z}) = \frac{1}{n}\sum(x_i, y_i, z_i), \tag{4.3.4}$$

$$\bar{\phi} = \text{atan2}(\bar{z}, \sqrt{\bar{x}^2 + \bar{y}^2}), \tag{4.3.5}$$

$$\bar{\lambda} = \text{atan2}(\bar{y}, \bar{z}). \tag{4.3.6}$$

4.3.3 Annotation of Movement Data

Annotation of movement data is an important step in a number of studies that link the surrounding environment of a species to movement behaviour. Acquiring the environmental variables at each relocation within a trajectory is performed so often that

online repositories of movement data are now allowing users to annotate their data before downloading it [DBW+13]. For example, the Movebank data repository has released the ENV-Data tool that allows users to annotate their data from a number of data sources, such as ASTER and MODIS. This system, however, does not support environmental datasets created by the user and requires a stable internet connection. To allow users to annotate their movement data with any number of environmental datasets, including those of their own creation, we implement an annotation system that can annotate movement data using all loaded environmental datasets with a chosen interpolation method.

Before annotation can be performed, the user is required to select a method of interpolating between datapoints. For spatial interpolation, nearest neighbour and bilinear interpolation is available. Nearest neighbour simply annotates the relocation with the closest datapoint for each environmental dataset. Bilinear interpolation uses the four surrounding points p_1, p_2, p_3, p_4 and their associated values v_1, v_2, v_3, v_4 to interpolate the desired value, V_u, at the point, P_u, using the following equations:

$$f_1 = \frac{x_2 - x}{x_2 - x_1} v_1 + \frac{x - x_1}{x_2 - x_1} v_2, \tag{4.3.7}$$

$$f_2 = \frac{x_2 - x}{x_2 - x_1} v_3 + \frac{x - x_1}{x_2 - x_1} v_4, \tag{4.3.8}$$

$$V_u = \frac{y_2 - y}{y_2 - y_1} f_1 + \frac{y - y_1}{y_2 - y_1} f_2. \tag{4.3.9}$$

For points that lie within two timestamps of a dataset, there are three methods to select the appropriate data; nearest neighbour, previous timestamp and interpolation. Nearest neighbour selects the timestamp of data that is closest to the desired datapoint. The previous timestamp method selects the timestamp that directly precedes the desired datapoint, as shown within Figure 4.1. This is useful for datasets

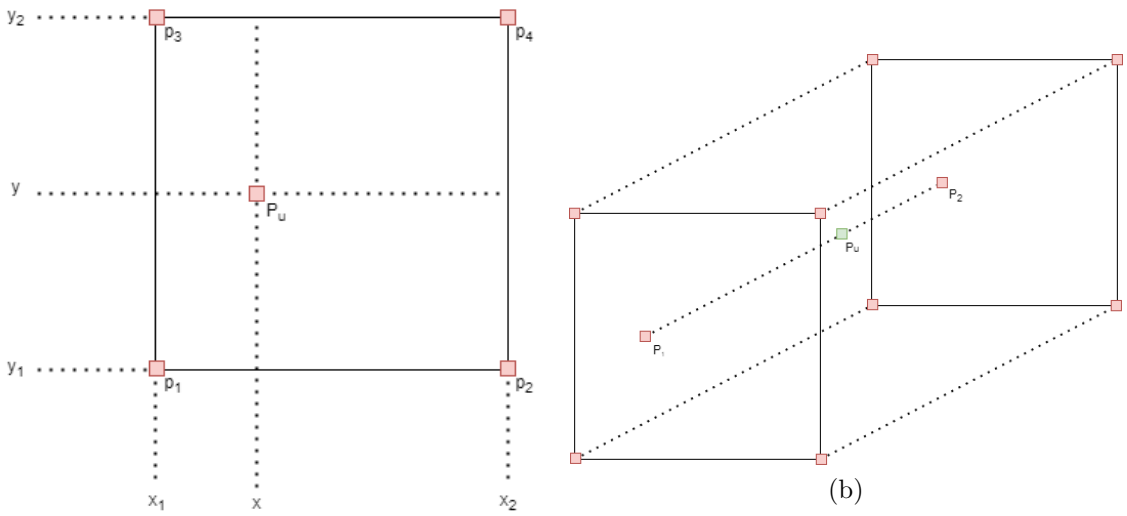


Figure 4.1: When annotating trajectory data with underlying environmental data interpolation can be applied. When this is desired, interpolation is first applied spatially (a). If a point lies within two timestamps, the spatial interpolation method is applied to both, and then a linear interpolation is applied between these two points (b).

that describe an average over a period of time, such as the MODIS average NDVI dataset. Finally, interpolation obtains the two timestamps and applies the chosen spatial interpolation method. Linear interpolation is then performed to obtain the desired datapoint. To avoid null data interfering with the annotation process, if any of the four surrounding points contain a null value denoting missing data, the interpolation will not be applied. The value returned will instead be the null value denoted by the file type of the underlying environmental dataset.

4.4 Event Detection

The identification of significant points within a trajectory has been investigated throughout literature [AAH+13, PBS+18, SAF+20, ESB16, HLZL20]. Each of these event detection methods has one or more parameters that can be adjusted, which can have a significant effect on the resulting events. During data driven exploration

Event Detection Method	Description	Literature
Metric thresholding	Uses a previously calculated metric selected by the user and checks for any values that are either above (or below) a certain threshold.	[SFA19]
Daily event detection	Determines the first and last relocation of each day to be events.	[SFA19]
Spatial thresholding Migration detection	The relocation that passes a user defined longitude (or latitude) threshold is determined to be the beginning of a migration. The relocation that passes a second user defined value is determined to be the end of migration.	[SAF+20, MGS+18]
Absolute displacement Migration detection	An extension of spatial thresholding where the start of a migration is then defined as the first day in which the animal moved the minimum daily distance that crossed the starting threshold, with the end of migration defined as the last day the animal moved the minimum daily distance after passing the second threshold.	[SAF+20, BCD17]
Squared Circular Standard Deviation (SCSD) turning point detection	A turning point detection algorithm that uses circular statistics to determine 'spikes' of the distribution of headings across discrete line segments.	[PBS+18]

Table 4.3: A table containing the five different event detection methods.

and analysis of movement data, the ideal parameters are often unknown and are obtained through a combination of expert knowledge and experimentation through data driven methods [WTB+20]. With the increasing sizes of datasets, these methods require longer computation times. While performing the computation once may have acceptable waiting times for computation to finish, the repeated computation required for experimentation can quickly become cumbersome. This delay can leave a disconnect between the results and the adjustment of parameters. To alleviate this, the performance increase given by GPGPU acceleration would allow for the rapid recalculation of these results, allowing for researchers to experiment with parameters. This section presents five GPGPU accelerated methods of detecting events; metric thresholding, daily event detection, absolute displacement migration selection, spatial thresholding migration selection and Squared Circular Standard Deviation (SCSD) turning point detection.

4.4.1 Metric Thresholding and Daily Events

The first method to detect events is metric thresholding. This method uses a previously calculated metric selected by the user and performs stream compaction to identify relocations when a certain threshold is met. This threshold can detect either points within the trajectory that either pass above or below the threshold. For example, as an initial analysis for the migratory pattern of a species, the user could calculate the turning angles for each point within a trajectory using a transform, reduce the turning angles to obtain the mean and standard deviation of the turning angles, and then threshold the turning angles based on these summary statistics to detect turning points of possible interest. Stopovers and regions of high foraging can also be detected by thresholding the FPT of a tagged animal. The radius of the FPT metric and the threshold can then be adjusted to allow for differing foraging periods to be highlighted. A threshold can also be applied to the step speed, identifying points that lie beneath a certain threshold could identify stopovers. While this method is basic in principle, in combination with other methods could highlight interesting behaviour.

Daily event detection places an event at the first and final locations of each day. To achieve this, stream compaction is performed that checks the timestamp of each relocation within a trajectory to see if the previous point occurs on a different day. This results in an array that contains the first and last point of each day, which are then transformed into events and returned.

4.4.2 Migration Selection

Migration phenology, the study of movement between breeding and non-breeding sites, continues to be a highly investigated area of movement ecology. As human

behaviour and climate change continues to transform the world around us, its effect on migratory timings becomes more substantial. To this end, the identification of migratory periods allows for researchers to further investigate these effects. Soriano-Redondo et al. have noted that while migratory timings are often identified in a number of different studies, the methods used can vary significantly, with expert knowledge playing a significant factor in their development and use [SAF+20]. To allow for researchers to identify migratory periods for large movement datasets, we implement two GPGPU accelerated migration detection methods highlighted by Soriano-Redondo et al; spatial thresholding and absolute displacement. With the results of these migration detection methods being particularly sensitive to parameter adjustment, the real-time detection of these points allows for researchers to adjust and investigate the effect of different parameters. While these methods lack the robust statistical backing of more complex methods, such as change point analysis or model fitting, recent literature has shown that these simpler methods have advantages over their more complex counterparts [CVB+20, SAF+20].

Spatial thresholding is implemented by first defining longitude and/or latitude thresholds that describe the beginning and ending of a migration. This is achieved by first applying a mask to the entire dataset, flagging the valid elements that cross the migratory threshold. Stream compaction is then used to reduce the dataset to contain only the points in which the threshold is passed. If they lie within the migratory threshold, the first and last points are then identified as migratory periods. If the trajectory needs to be segmented into differing migratory periods, an inclusive sum scan can be performed. This annotates the trajectory such that each migratory period will have an ID associated with it, allowing different migratory periods to be visualised.

Absolute displacement is a method of identifying migratory periods similar to

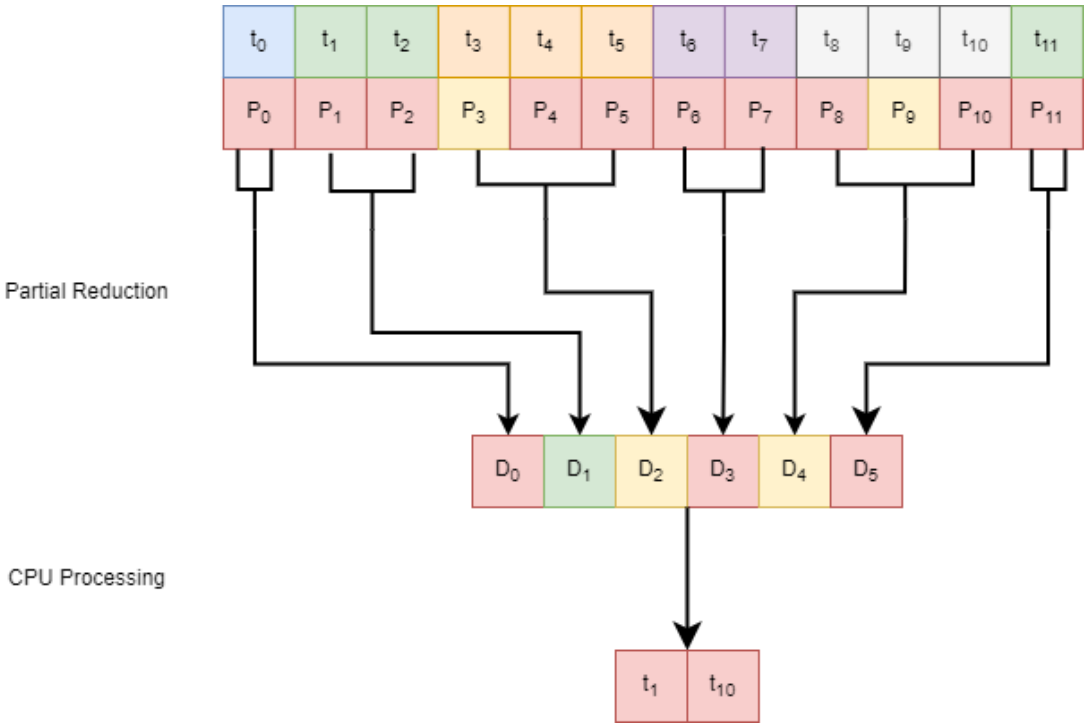


Figure 4.2: The main step of absolute displacement migration detection. Each relocation is first transformed into a day structure containing a value for the first and last relocation of a day. These are then partially reduced, such that if a relocation occurs on the same day with its neighbour it is reduced to a single value together with the first and last timestamps updated. A flag is also set if a relocation crossed the threshold defined by the user. This results in an array that contains each day and if the threshold was passed in that day. Due to the drastically decreased size of the array this is then transferred to the CPU for sequential processing. Here we sequentially check if days previous to the start of migration (in this case D_2) or days after the end of migration travel further than the specified distance. Here, D_1 is the day proceeding the migration and the minimum distance is travelled, so t_1 is returned as the start of the migration.

spatial thresholding. Instead of just having parameters for the starting and ending thresholds, absolute displacement also takes the minimum daily distance as a parameter. The start of a migration is then defined as the first day in which the animal moved the minimum daily distance that crossed the starting threshold, with the end of migration defined as the last day the animal moved the minimum daily distance after passing the ending threshold. To find these points on the GPU, each point within the trajectory is transformed into an intermediate format that contains the timestamp, its associated position, and if the point crosses the threshold. A partial reduction is then performed, using the timestamps as a key to partially reduce to one value per day. This returns a struct containing the first and last timestamps of each day, with their associated positions, and if the threshold was crossed that day. With the number of points now vastly reduced, one for each day, it is transferred back to the CPU where the trajectory is sequentially processed to identify if any of the days, before or after the initial migration thresholds, have a distance moved more than the minimum daily distance parameter. Again, similar to spatial thresholding, if an annotated trajectory is desired, rather than just the migration dates, a sum scan can be performed. This is shown in Figures 4.2 and 4.3

4.4.3 Turning Point Selection (SCSD)

With the efficacy of tagging technology improving significantly over the last decade, techniques that process this higher relation data are needed. Potts et al. have noted that many existing techniques that analyse movement data were designed for lower temporal resolutions [PBS+18]. To account for this, they implement a turning point algorithm that uses the high resolution data available to detect turning points from heading, rather than position. The formulae for their Squared Circular Standard Deviation (SCSD) implementation is covered in Chapter 2.6.2. While Potts et al. have

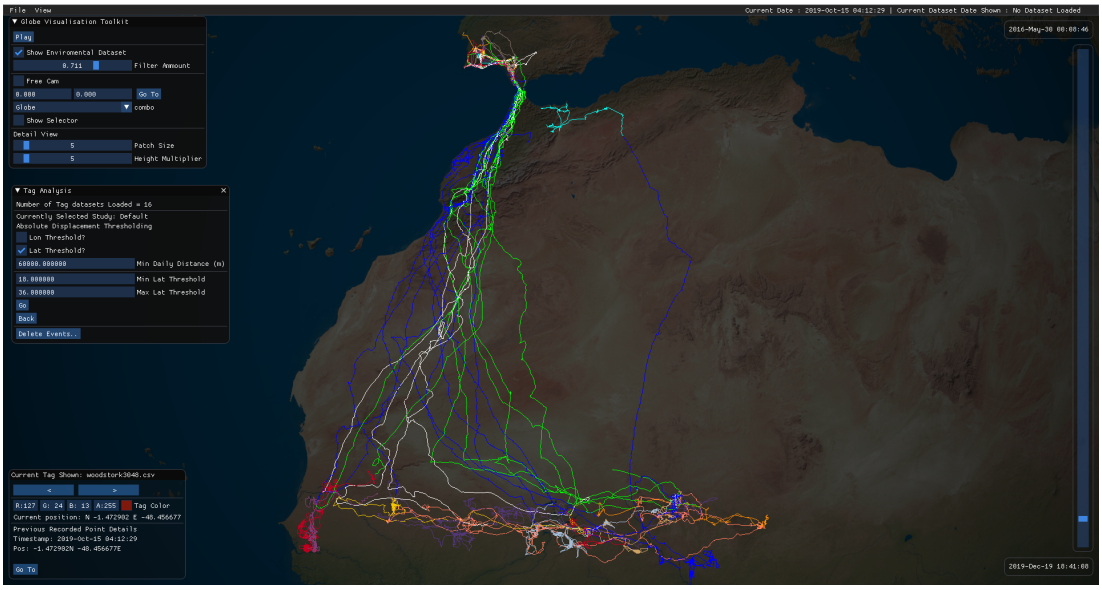


Figure 4.3: The results of absolute displacement migration detection for a group of while storks between 2016 - 2019. Here we see three distinct migrations, with outbound migrations highlighted in blue, return migrations in green and the final outbound migration in white.

optimised their implementation to allow for large amounts of data to be processed on the CPU, there is a limit to sequential processing. They state in their paper that the algorithm "can process 100,000s of data points in a few seconds" and that "even tracks of a billion locations (40Hz for a year) would be analysable in only a few hours" [PBS+18]. We suggest that this algorithm, with some minor changes, is ideal for the throughput nature of GPGPU computing, allowing for large datasets of high resolution data to be processed in a timely manner.

To begin, the cosines and sines of all headings are calculated using a parallel transform. These two arrays are then passed to a kernel that calculates the SCSD for each point within the trajectory with a given window size. Due to each thread calculating the SCSD for each point in turn, all access to the sine and cosine arrays are coalesced. A sum reduction is then used to calculate the average SCSD for the entire trajectory. The average SCSD is then used as a threshold in a stream compaction

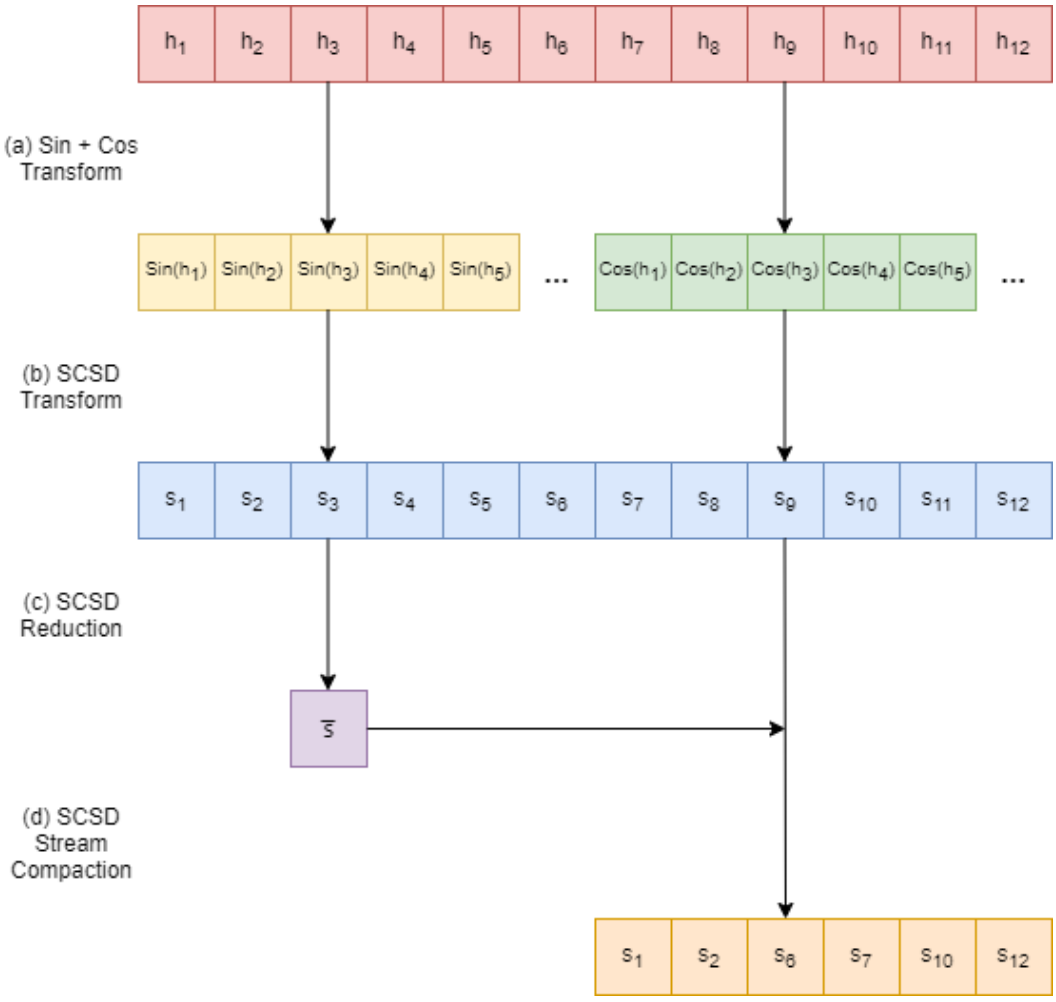


Figure 4.4: The main steps of detecting turning points using the Squared Circular Standard Deviation (SCSD) method by Potts et al. [PBS+18]. (a) The headings are first transformed into sine and cosine arrays to avoid recalculation during the transformation (b) of the sine and cosine arrays into SCSD for each relocation. A parallel sum reduction (c) is then performed to calculate the average SCSD $\bar{s}$ over the entire trajectory. $\bar{s}$ is then used in a stream compaction step to obtain each of the points that pass this value, which is then compacted again to obtain the centre point of each region that passes over the mean SCSD value. Sum scans are then performed on the Sine and Cosine arrays for post processing to remove any points that do not pass the user specified threshold.

step that returns an array containing the points within the trajectory that pass the average SCSD, either passing above or below. This compacted array is then reduced further to calculate the centre index between each turning point pair, returning the final turning points. An inclusive sum scan is then performed on the sin and cosine arrays, which are used for post processing to remove any turning points where the change in direction is less than the given threshold. To reduce the amount of data needed on the host memory for post processing, a permutation iterator is used to only copy the turning points from the sum scans of the sin and cosine arrays. This is shown within Figure 4.4.

4.5 GPU Accelerated Resampling of Movement Data

Due to the often unreliable nature of data collection, and with various analytical methods requiring a regular sampling interval, resampling of movement data is commonly performed on movement datasets. As covered in Chapter 2.6.3, there are a number of different methods in literature to obtain a regular sampling interval throughout the trajectory. Here, we implement a parallel resampling algorithm that allows for a number of resampling methods to be implemented using the power of GPGPU computing. We then apply this method to two resampling methods common in movement ecology, linear interpolation and Catmull-Rom splines. The method takes three parameters, the initial trajectory $p_0, ..., p_{n-1}$, the initial timestamps $t_0, ..., t_{n-1}$ and the desired sampling interval Δt. First, the desired timestamps are calculated by transforming a counting iterator (in essence, an array containing the values $0, ..., N-1$) and multiplying each element by Δt. This array $T_0, ..., T_{N-1}$, is then merged with the initial timestamps, with a mask being applied using merge_by_key such that this results in two arrays, $C_0, ..., C_{N+n-1}$ the combined timestamps from the initial trajectory and the desired timestamps, and $M_0, ..., M_{N+n-1}$ containing the mask, a 1 if the

timestamp is from the original trajectory and a 0 otherwise. An inclusive sum scan is then performed on the mask array M. This results in an array that contains the index of the next point within the initial trajectory for each relocation within the desired trajectory. Stream compaction is then performed on this array using the initial mask M to remove any indices of the original trajectory. This stream compaction step can also be used to filter out unwanted relocations. For example, using the index of the approaching relocation from the original trajectory, the time difference between the current timestamp and the previous timestamp from the original trajectory can be calculated. If the time difference is too great, the timestamp can be removed. This final array contains an index of the next relocation within the initial trajectory. This can then be used with thrust permutation iterators to perform the method of resampling desired in parallel.

The first of the resampling methods we implement is linear interpolation. This is implemented by sending four permutation iterators to a thrust transform function. This results in the previous and following recorded relocations and timestamps being available to each thread quickly in parallel. A linear interpolation can then be performed using these values and the desired timestamp for each relocation. To implement a Catmull-Rom spline resampling, the same data is used as the linear interpolation, with the addition of the two extra control points. These are obtained by having another pair of permutation iterators, giving each thread access to the two elements directly preceding the relocation, and the two relocations and timestamps following each point. The Catmull-Rom implementation is calculated using the glm::catmullRom function. While we only show resampling using these two methods, other methods (such as Kinematic interpolation by Long [Lon16] or Bezier curves) can be implemented using this method. Any other variables needed can be obtained using the array containing the index of the next recorded point to obtain any information

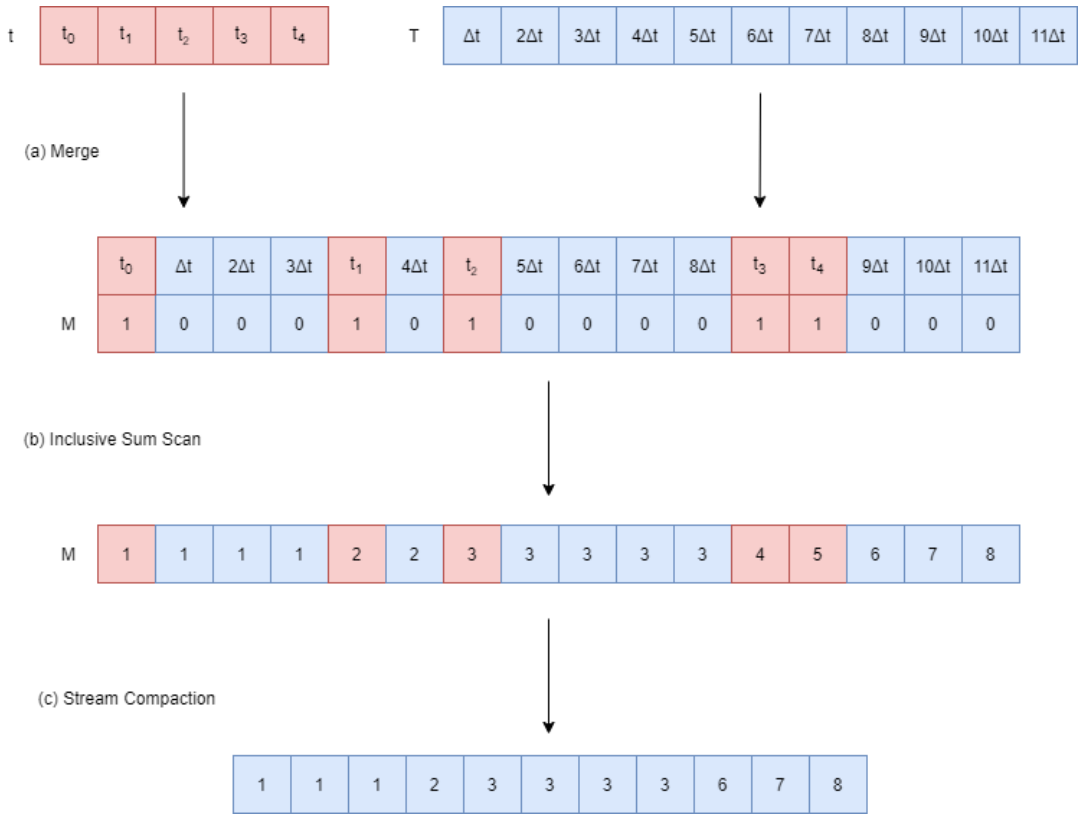


Figure 4.5: The three main steps of resampling movement data on the GPU. (a) First the timestamps generated and the timestamps from the initial trajectory are merged together, with a mask being applied. (b) An inclusive sum scan is then performed, denoting the index of the approaching relocation that lies within the initial trajectory. (c) Stream compaction is then performed using the initial mask array to remove any points from the original trajectory and any invalid points (such as time step being too large to sensibly apply interpolation). This final array can then be used to implement a resampling method of choice.

N	GPU (ms)	CPU (ms)	Speedup
37317	29.3064	843.38	x29
49114	35.0629	1404.72	x40
59996	52.6564	1874.19	x36
66973	52.6653	2617.02	x50
81856	95.744	3912.42	x41
105243	151.131	6416.74	x42
184175	349.306	20809.7	x59

Table 4.4: The timing results for first passage time calculation, with an input radius was 5Km. The CPU times are taken from the adehabitatLT package.

needed. This is shown within Figure 4.5.

4.6 Results

The data used for all testing was collected by my secondary supervisor Aldina Franco from 2016 - 2019 and compiled on Movebank [FCRA19]. To obtain differing sizes of trajectories for testing, a subsample of differing trajectories were used to test datasets of smaller sizes. The effect of a higher sampling interval was also tested by resampling a trajectory to a higher temporal resolution. To test the SCSD turning point algorithm, the test data supplied within the supplementary materials was used [PBS+18]. All testing was performed 100 times, using a system running Windows 10 with an Intel i7-8700 CPU @ 3.2GHz, 32Gb RAM and an NVIDIA Titan XP, and an average runtime is presented. All timings presented for GPU methods include sending relevant data to the GPU, the kernel execution, and then subsequent transfer back to CPU memory. It should be noted that some methods do not require this transfer to and from host memory when utilised, as the position data for each trajectory is already located on the GPU for visualisation. We present these methods with host to device transfers so that these timing can be more easily replicated in isolation.

N	GPU (ms)
1661	0.80
41726	0.81
70839	0.80
280738	1.25
422771	1.35
431377	1.48
433375	1.60
479980	1.45
522426	1.77
552523	1.90
554465	1.79
889657	2.23
983429	2.51
1029193	2.29

Table 4.5: The time required to calculate the summary statistics of a previously calculated movement metric. In this case, the standard deviation, mean, minimum, maximum, variance, median and skew with a single reduction.

4.6.1 Metrics

Due to the nature of most metric calculations, speedup was observed. Due to metric calculation lending itself exceptionally well to the nature of parallel processing, a trajectory containing 589,606 relocations only requires 14 milliseconds to calculate the distance for each step. Other stepwise metrics, such as time step and step speed, require a similar amount of time to process. While these speedups are useful when dealing with exceptionally large datasets, these transformations are typically performed as part of a more involved process, such as event detection. Where GPGPU acceleration becomes particularly apparent when calculating metrics that process multiple relocations, such as FPT. As can be seen from Table 4.4, there is a significant speedup when compared to the CPU implementation within the adehabitatLT package. As the sampling interval decreases or the size of the radius is increased, more points lie within the threshold. This causes the computational complexity to tend

N	GPU (ms)
1840	0.45
25377	0.80
35311	0.87
39088	0.87
41618	0.89
60651	1.036
109284	1.27
187159	1.60
230811	2.013
281848	2.49
348284	2.70
371216	3.38
589606	6.06
686129	7.51

Table 4.6: The computation time required to threshold a metric for a specific value to determine event points. For example, this can be used in combination with metric calculation and summary statistic reduction. For example, thresholding step speed for the identification of flight modes.

towards $O(n^2)$. This makes the computation time of FPT calculation particularly sensitive to sampling interval and search radius. The adehabitatLT package has also been implemented using RCpp. This requires the conversion of a trajectory in R to an array or vector in C++. This added conversion requires a significant time for larger datasets. This results in a speedup of up to 50 times. With the GPU accelerated method requiring an order of magnitude less time, the search radius of the FPT calculation can be adjusted and the results examined to investigate the foraging levels at differing scales.

Via the use of parallel reduction, summary statistics of these metrics can be calculated very quickly. As shown in Table 4.5, calculating the standard deviation, mean, and minimum and maximum of 1 million step distances requires 2.3 milliseconds, allowing for the summary statistics of an entire study to be calculated quickly for use in more advanced analysis. Furthermore, partial reduction of trajectories into daily

N	Time (ms) GPU	Time (ms) CPU	Speedup
42665	1.57	2154.40	x1371
63998	2.57	3227.52	x1255
79997	2.80	4854.42	x1732
106663	3.63	8355.35	x2296
159994	4.97	19658.58	x3948
213325	6.11	35880.81	x5870
255989	7.20	52472.32	x7288
319987	8.51	85394.68	x10023

Table 4.7: The computation time required for the identification of the beginning and end of migrations using spatial thresholding. We compare the R code from the supplementary material from Soriano-Redondo et al [SAF$^+$20] labelled as CPU and our new GPGPU based method.

distances requires 16 milliseconds for a trajectory containing 589,606 points.

4.6.2 Event Detection

As can be seen from Table 4.6, with the rapid calculation of metrics covered in the previous section, the use of stream compaction to detect events allows for real-time investigation movement metrics. For a trajectory containing 589,606 points, the calculation of step speed and thresholding these values, such that all relocations below flight speed are marked as events, requires 6 milliseconds on the GPU. This method can be combined with the reduction of metrics to calculate summary statistics, with the standard deviation and mean used to create a desired threshold. The detection of the first and last points of each day via partial reduction requires a similar time to the calculation of daily distances, due to using a similar reduction method that instead returns the indexes of the first and last points of each day, instead of the distance between the two.

N	Time (ms) GPU	Time (ms) CPU	Speedup
42665	3.421	125.12	x36
63998	5.585	116.83	x20
79997	5.878	219.77	x37
106663	7.326	114.18	x15
159994	17.41	275.85	x15
213325	13.56	128.64	x9
255989	18.23	132.48	x7
319987	22.16	213.48	x9
639972	38.37	311.56	x8
1279944	79.46	721.34	x9

Table 4.8: The timing results for absolute displacement migration detection. As can be seen, the speedup trends towards 9 times when compared to the R code from Soriano-Redondo et al [SAF+20].

Migration Detection

Here, we compare our GPU accelerated migration detection implementation with the CPU method obtained via the supplementary material within the paper by Soriano-Redondo et al 4.8. As can be seen from Tables [SAF+20] and 4.7, migration detection has seen a significant increase in performance when compared to their original CPU counterparts. Absolute displacement thresholding on the GPU has seen a speedup of 35 times at lower datasets sizes, and trending towards 9 times faster for larger datasets. The main speedup for this method is the GPU reduction step, where each position is reduced to days. This results in better timing when the resolution of the dataset is increased, with more points for each day, rather than just increasing the duration between the start and end dates of the trajectory. The spatial migration thresholding has seen the largest performance increases, with a speedup of up to four orders of magnitude when compared to the R code by Soriano-Redondo et al [SAF+20]. Table 4.7 shows that the GPU method scales significantly better than the CPU method in R, with the CPU method requiring 85 seconds to process a trajectory

N	Time (ms) GPU	Time (ms) CPU	Speedup
73989	2.0	80.2	x40
144000	1.9	120.5	x63
216000	2.1	270.7	x129
432000	2.8	410.4	x147
2219670	16.8	2450.2	x146
8878680	35.6	13180.1	x370

Table 4.9: The computation time required to determine turning points using the Squared Circular Standard Deviation (SCSD) method by Potts et al. The code used for the CPU method was obtained via the supplementary materials [PBS+18].

containing 319,987 points, and the GPU method able to process the same in 8.5 milliseconds. This is due to the R code simply sequentially checking each individual point for passing the threshold within a loop. With R loops being notoriously slow, due to being an interpreted language, this has a significant effect of performance, whereas the GPU method uses stream compaction. Migration detection is sensitive to parameter adjustment, from both the thresholds and the minimum daily distance. This significant reduction in computation time allows for interactive adjustment of these parameters without the disconnect of having to wait for results. This is particularly prevalent as our studies increase in size to containing millions of datapoints.

Turning Point Detection (SCSD)

The turning point detection algorithm designed by Potts et al. was designed for very high resolution data [PBS+18]. It has therefore been optimised for rapid processing on the CPU. Here we compare the CPU implementation provided within the supplementary materials of their paper. As shown in Table 4.9, the calculation of SCSD lends itself well to the throughput nature of GPGPU computing. A trajectory containing 8,878,680 points, with a window size of 40 and a turning threshold of 30°, required 35 milliseconds to process. This is two orders of magnitude faster than the CPU method, requiring 13 seconds. With a sampling interval of 40Hz, this data would cover just

Sampling Interval (s)	Output size	Points Added	Time (ms)
900	91484	-444099	17.27
600	137226	-398357	18.63
480	171533	-364050	19.67
360	228710	-306873	22.21
300	274452	-261131	24.47
120	686129	150546	38.27
60	1372257	836674	60.58
30	2744514	2208931	105.61

Table 4.10: The computation required to resample a single trajectory containing 535,583 points to differing regular sampling intervals. If the Points added is negative it indicates a down-sampling, when it is positive it is being up-sampled.

over two and a half days. With migratory periods typically spanning months or even years, these trajectories would contain billions of datapoints. Potts et al. state that this "would be analysable in only a few hours" [PBS+18]. With GPU acceleration, this computation time would be significantly reduced. However, it should be noted that memory may become an issue at these dataset sizes. The memory requirement for a trajectory that spans one year at 40Hz would require just over 10GB of space. While this is feasible for most systems to contain in RAM, to fit the entirety of the trajectory onto the GPU, alongside other system critical processes and visualisations, is only feasible with a high performance GPU. Movement datasets of these sizes and resolution are rare, however.

4.6.3 Resampling of Movement Data

The throughput nature of GPU computing lends itself well to the resampling of movement data. To assess the performance of the methods used, we first compare the computation time required to resample a single trajectory to multiple sampling intervals (Table 4.10). We then compare the computation time required to resample multiple trajectories of differing sizes (Table 4.11). First, a white stork trajectory containing

N	Output Size	Points Added	Time (ms)
60651	374317	313666	16.89
25377	563695	538318	22.01
42008	575169	533161	22.17
29723	577833	548110	21.67
109284	639973	530689	26.57
35311	696567	661256	27.39
39088	699345	660257	26.28
46898	715918	669020	26.98
37317	736697	699380	27.24
41618	739287	697669	28.60
43373	742431	699058	29.68
230811	1186209	955398	47.75
589606	1311239	721633	59.20
535583	1372257	836674	62.59

Table 4.11: The computation time required to resample each trajectory within a single study. The study used was the migratory data of white storks.

535,583 points, recorded between Jan 13th 2017 and Aug 24th 2019, was resampled to a regular interval. With an interval of 30 seconds, the resampling required 105 milliseconds to output a trajectory containing just over 2.7 million datapoints. Both the Catmull-Rom spline and linear interpolation required similar times, with the kernel execution of Catmull-Rom requiring approximately 2ms longer for the same output. This shows that while the Catmull-Rom spline kernel is more complex than linear interpolation, the algorithm is mostly dependent on the size of the output trajectory. As the size of the output trajectory increases, the number of resampling operations and memory operations increases. As is shown in Table 4.10, downsampling is much faster, requiring 17 milliseconds to output 91,484 points. This is due to the lower amount of resampling and memory operations performed.

As is more likely in a real world study, Table 4.11 shows the effect of up sampling the temporal resolution of the initial trajectory, thus increasing the size of the output array. These trajectories describe the migration of white storks for the years

of 2016 - 2019, varying in length from a single migration to multiple years. Here we can see that the output trajectory size effects the computation time. A trajectory initially containing 60,051 points resampled to 374,317 points required 16.9 milliseconds, whereas another trajectory initially containing 535,583 resampled to 1,372,257 points (approximately 3.7 times the amount) requires 62 milliseconds (3.7 times the execution time). Regardless, all resampling was completed in under 70 milliseconds. This real time result allows for researchers to resample their data at a multitude of resolutions, using multiple interpolation or spline methods to investigate their effects on the data.

4.7 Conclusion

Due to the nature of spatio-temporal data, with each point discretely describing a relocation within a trajectory, it lends itself exceptionally well to GPGPU computing. Some common metrics and methods are 'embarrassingly parallel' with a natural adaptation to parallel programming, such as measuring the speed and turning points of each relocation, where each relocation can be processed individually, reading only the current and previous relocations. Other more complex methods, such as turning point detection, can also be adjusted to allow for the use of GPU hardware. As the sizes of movement datasets continue to increase, both from more tags being deployed and tagging efficacy allowing for more detailed trajectories with higher temporal resolutions, the need for scaleable solutions to process this big data is needed.

Here we have implemented a number of common early processing methods used by movement ecologists when investigating the migratory behaviour of long lived birds, such as calculating movement metrics that describe each relocation within the trajectory. These commonly used movement descriptors, such as step speed, step length and turning angles, can be used in other more involved data driven methods. We have

also facilitated the real-time detection of event points that describe regions of interest within a trajectory. These events can be selected by thresholding movement metrics to highlight relocations. For example, thresholding first passage time can highlight regions as an indication for foraging behaviour [FT03]. We have also implemented GPGPU accelerated migration detection, based on the methods described by Soriano-Redondo et al [SAF+20]. This allows for researchers to quickly identify migratory regions in large amounts of movement data. Finally, we have implemented GPGPU accelerated resampling of movement data, implementing both bilinear interpolation and Catmull-Rom splines. The resampling method can also be extended to implement other resampling methods. This method also allows for selective resampling, with the adjustable stream compaction step able to limit the amount of resampling performed. We then tested these methods on white stork data for the years 2016 - 2019 [FCRA19] and have shown that real-time calculation of metrics, event detection and resampling is possible.

While networked clusters and distributed computing solutions allow for larger dataset sizes to be processed [GWD20], we argue that there is still merit in processing on a single workstation, with methods that are easier to be reproduced by a single researcher. We propose that researchers begin to utilise their GPUs' power to process more data, more quickly on a single workstation than with sequential CPU-based methods. With the real-time processing of movement data, we hope that this new interactivity will enable researchers to glean insights into their data faster and easier than before. While GPU methods for a single workstation are unable to handle the sheer amount of data that a networked cluster can, we believe that there is a middle ground that is achieved, with gigabytes of data able to be processed by a single workstation in a real-time environment, with larger terabyte, or even petabyte datasets being processed by networked clusters.

Chapter 5

GPGPU Accelerated Flow Diagrams and Spatially Filtered Space Time Cubes

5.1 Introduction

Exploration and investigation mark an important prerequisite for the development and adjustment of hypothesis and data driven models, with development of visualisation methods allowing for more effective communication of information between researchers and engagement of public outreach. In today's 'big data' environment the sheer volume of data can cause problems in the creation of visualisations [GG13]. Pre-rendered visualisations are available, but these have limitations. For example, Schwalb-Willmann et al. [SW19] states that 'rendering an animation of 500 frames with 25 frames per second, resulting in an animation duration of 20s, takes approximately 25–35 min' for their recent R package moveVis. These visualisations, while animated, restrict the users view to a predetermined point of view without re-rendering, requiring repeated long computation times if other views are desired. This leaves a disconnect between the adjustment and resulting visualisation.

An overabundance of data also creates a number of issues not tied to the computational complexity of the visualisation. Occlusion of data is also an issue for large

datasets, where multiple data points overlap, causing some parts of the data to become obscured. The high rate of data change also causes issues when monitoring large amounts of data [GG13]. A number of aggregation techniques have been implemented to solve this issue, with density maps very prevalent in literature [DV10a, BND+20]. More involved aggregation techniques are beginning to be developed, allowing for the visualisation of not only where movement is happening, but also what direction. Andrienko [AA11] developed a method of aggregation that creates a number of flows between 'characteristic points'. Recent developments by Graser et al. [GWD20] 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 the adjustments described within this chapter to the original methods used by 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 datapoints found in today's ecological movement datasets. We adapt the sequential method of Andrienko [AA11], making it more suitable for the GPU architecture. This allows for interactive adjustment of parameters used in creating the aggregate flow map, removing the disconnect of adjustment and results that occurs when long computation times are required. We then apply this to space time cubes, allowing for further levels of generalisation interactively available to the user. 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.

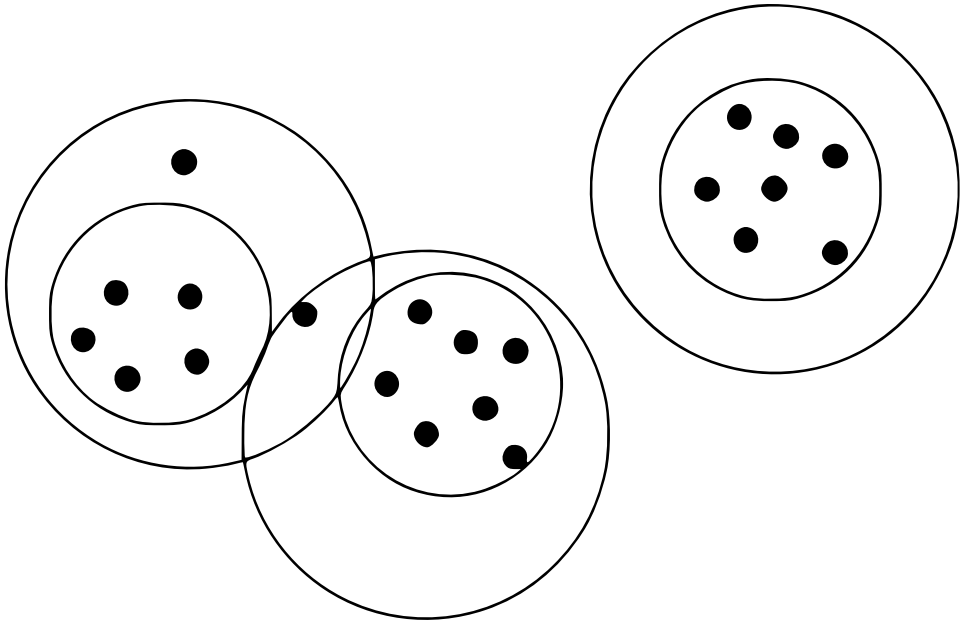


Figure 5.1: An example of the Canopy Clustering Algorithm by McCallum et al [MNU00]. As can be seen, a point is selected from the dataset and the points within the loose distance are obtained. The centroid is then calculated and the points within the loose distance are removed. This is repeated until no more points remain.

5.2 Flow Map Calculation

Much like the method implemented by Andrienko and Andrienko [AA11], 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 that the tag event detection from Chapter 4 has already taken place and these points are used as 'characteristic points'.

5.2.1 Clustering of characteristic points

With characteristic points identified, the clustering of these points can begin. These centroids act as nodes, with flows occurring between them. It is here that the level of generalisation is dictated, with clusters of a desired radius being created that drastically effect the level of detail of the resultant flow map. The initial method

used by Andrienko and Andrienko is covered in Section 2.5.2, but some of the issues addressed within this chapter are highlighted here. The main issue with the current clustering method used by Andrienko is the continuous recalculation of centroids. As each point is inserted into clusters using the regular grid, the centroid needs to be recalculated with this new point. While the search for the closest centroid is optimised with the use of a regular grid, this recalculation can cause significant computational complexity. This is especially apparent during the worst case, where all characteristic points are grouped within the same centroid, causing a time complexity of $O(n^2)$.

Another issue stems from the construction of the regular grid used to optimise clustering. The algorithm contained within Andrienko and Andrienko's 2011 paper [AA11] contains no reference to the coordinate system used to create the regular grid. It is assumed that the method is performed using standard GPS coordinates of longitude and latitude. While this is suitable for smaller regions, issues begin to appear at a global scale. Without reprojection of coordinates to an equal area projection the initial method breaks down as points reach the poles and cause grid cells to become distorted. Reprojecting the points to the Albers equal-area conic projection could solve this issue, however this itself is an expensive operation and would require significant processing for millions of points.

To avoid the drawbacks associated with the use of a regular gird, while also optimising for the large number of points, the canopy clustering algorithm by McCallum et al. [MNU00] is applied using GPGPU acceleration. Originally designed as a pre-clustering algorithm for use with large, high dimensional datasets, its use here allows for the efficient clustering of large amounts of data while also adjusting the radius of the resulting clusters. This allows for adjustment of the level of generalisation. Pseudo-code is shown within Algorithm 1 and an example is shown within Figure 5.1. There are three inputs: the dataset to be clustered, the loose distance $T1$

and the tight distance $T2$, with $T2 < T1$. These distances are then used to create canopies over the dataset as follows. Beginning with the entire dataset to be clustered, a selected datapoint becomes the centre of a new canopy (this can be done either randomly, or simply the next point in the dataset). For each point remaining within the dataset assign it to the canopy cluster if the distance to the first point is less than $T1$. Furthermore, if the distance is less than $T2$, remove it from the dataset. Repeat the previous steps until no more points remain within the dataset.

Algorithm 1 Canopy Clustering CPU

```
1:  procedure CanopyClusterCPU(P, T1, T2)
2:      C = ∅                                          ▷ The set of canopies.
3:      while |P| > 0 do                               ▷ While points remain in the dataset
4:          S = ∅
5:          p_c = P[0]                                 ▷ Next point as the start of canopy.
6:          for all p ∈ P do
7:              if distance(p,p_c) < T1 then
8:                  S = S ∪ p                          ▷ Add point to canopy.
9:                  if distance(p,p_c) < T2 then
10:                     p ∉ P                          ▷ Remove point from dataset.
11:         C = C ∪ S
```

This method has some advantages over the initial implementation used by Andrienko [AA11]. First, each centroid needs only be calculated once. This avoids the recalculation of the centroid that occurs whenever a point is added to a cluster. Furthermore, due to the method relying only on distances rather than grid cells, the only conversion of coordinates needed is for centroid calculation. This is done by converting to ECEF, averaging the points then converting back. The distance calculations can also further be optimised by the use of a faster distance calculation for the initial comparison to the loose distance $T1$, which will be performed on the entire dataset. A more accurate but costly distance calculation for the comparison to the tight distance $T2$ can then be used. With adjustment this method also lends itself exceptionally well to the architecture of GPUs and parallel processing. It is this method that is

used to calculate the clusters, and thus centroids, of the flow map.

The GPU accelerated method is comprised of three steps: The canopy gather step, centroid reduction and point removal. For use in the creation of flow maps we are not interested in the specific members of each canopy, only the centroid. This allows for the removal of points from the dataset once processed without explicitly assigning them to a centroid.

To implement this, two pointers are used to track the creation of centroids and points removed by the dataset, the $P1$ and $P2$ pointers. They are both initialised to be the first point in the dataset. The distance for each point from the value at the $P2$ pointer is calculated and stream compaction is used to partition the values as such that all points that are within the loose distance precede all others. Due to the more numerous points within this step a less accurate but faster Haversine distance metric is used. The $P1$ pointer is then placed at the end of the points within the loose distance. This gathering operation obtains all points that will make up the canopy. It is here that the centroid calculation takes place. A parallel reduction is used to calculate the centroid of all points within the $P1$ and $P2$ pointers. The centroid is calculated by first converting all points to ECEF coordinates (Earth-Centred Earth-Fixed), covered in Equations 4.3.1 - 4.3.3. The sum of all points is then calculated during the reduction and the centre point of the ECEF coordinates are calculated. The centroid is then converted back to longitude and latitude. Another partition is then performed once more on all the points within the $P1$ and $P2$ pointers with all points within the tight distance preceding the points within the loose distance. It is here that a more expensive distance algorithm, such as the Vincenty distance, is used. The $P1$ pointer is then placed after the last point within the tight distance, removing them from the dataset. The process then repeats until $P1$ and $P2$ are at the end of the dataset. This method of implementation allows for control of not only

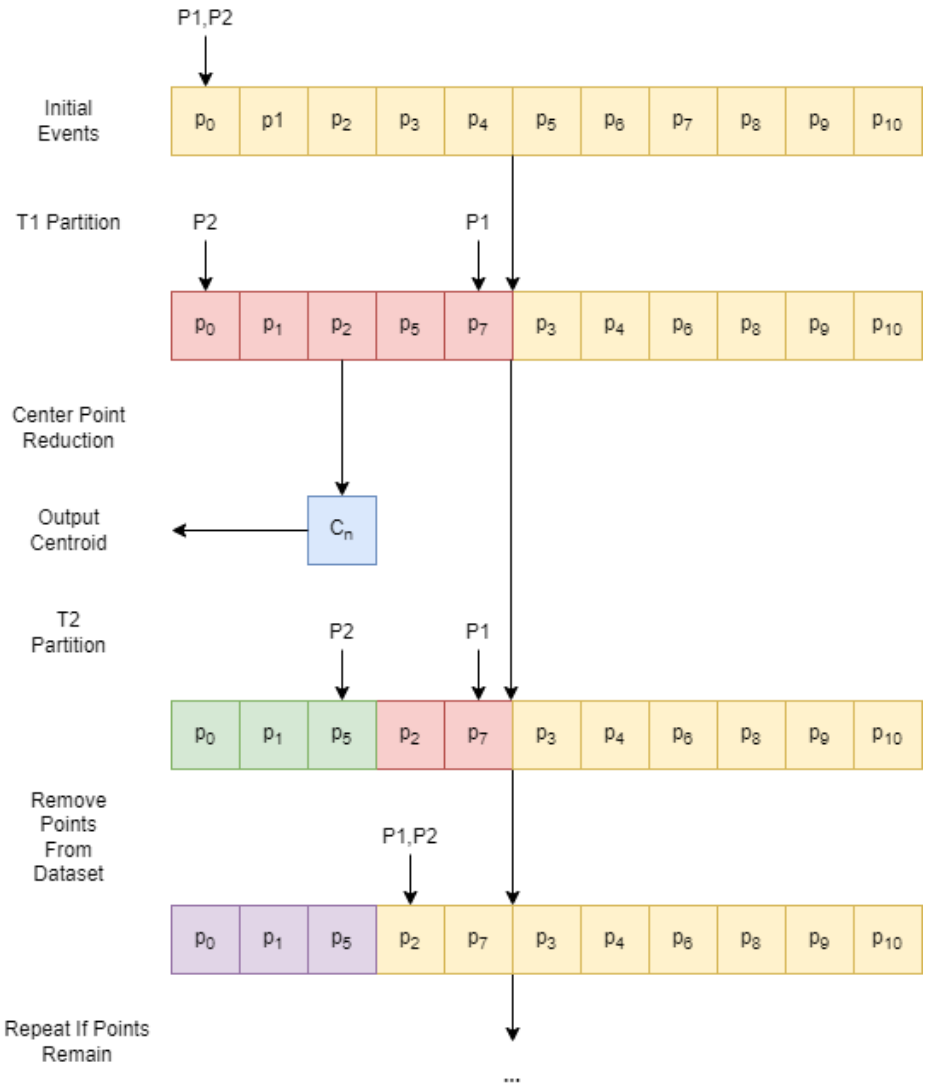


Figure 5.2: The steps required to calculate the centroids of a given set of events $p_0, ..., p_n$. First, the T1 partition is applied to obtain the events that make up the cluster which are then used in a parallel reduction to obtain the centroid. Next the T2 partition is applied to these event points to remove points from the dataset that are within the tight distance. This is repeated until no points remain.

the number of centroids via the adjustment of the $T2$ (tight distance) parameter, but also the effect of other outside points with the adjustment of $T1$ (loose distance). This is shown in Figure 5.2 and Algorithm 2.

Algorithm 2 Canopy Clustering GPU

```
1: procedure CanopyClusterGPU(P, T1, T2)
2:     C = ∅                                          ▷ The set of centroids.
3:     P_1 = 0
4:     P_2 = 0
5:     while P_1 ≠ |P| and P_2 ≠ |P| do      ▷ While points remain in the dataset
6:         p = P_2 + 1                              ▷ Start with next available point.
7:         P_1 = partition(P_2, |P|, T1, p)   ▷ Place all points within T1 before others.
8:         c = reduce(P_2, P_1)        ▷ Calculate centroid from valid points within T1.
9:         C = C ∪ c                                          ▷ Add centroid to set.
10:        P_2 = partition(P_2, P_1, T2, c)   ▷ Place all points within T2 before others.
```

5.2.2 Trajectory segmentation

With the clustering of characteristic points now complete, the segmentation of trajectories can begin. 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 each trajectory is then obtained. The trajectory is then represented as a sequence of cells visited, which 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. More detail is given in Chapter 2.

Instead of implementing a Voronoi diagram, we make use of GPU parallel primitives to accelerate distance measurements. To check for transitions between centroids each trajectory is processed in turn. 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

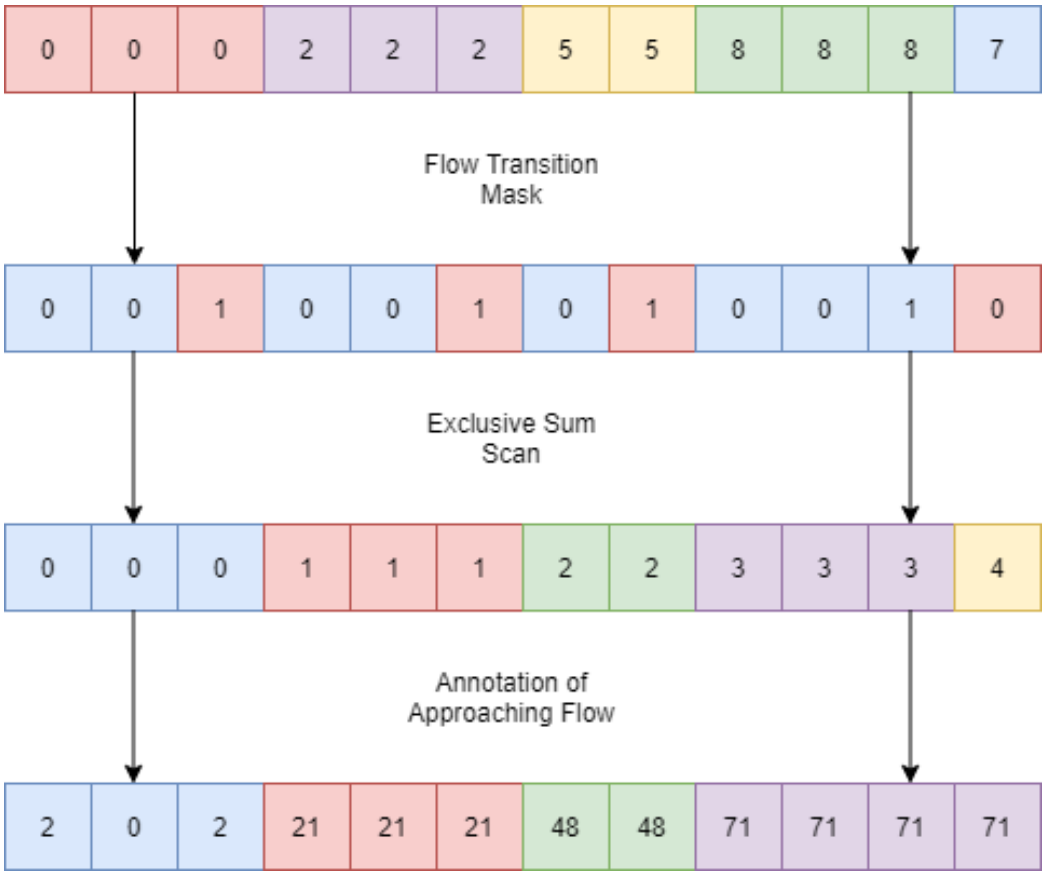


Figure 5.3: The main steps of trajectory segmentation and annotation. First, the closest centroid for each relocation in the trajectory is found. A mask is then applied identifying flow transitions. A parallel stream compaction is then used to extract the flow transitions followed by an exclusive sum scan to annotate the trajectory with the index of the approaching flow transition for visualisation. A permutation iterator is then used to annotate the trajectory with the approaching flow.

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 array, which we refer to as the annotated visits array, can then be used to calculate statistics of each visit to a centroid using segmented reduction. For example, the total time for each visit can be calculated by reducing each visit to its minimum and maximum timestamp elements, which can then be used to calculate the total time spent within that centroid. Total distance moved can also be calculated

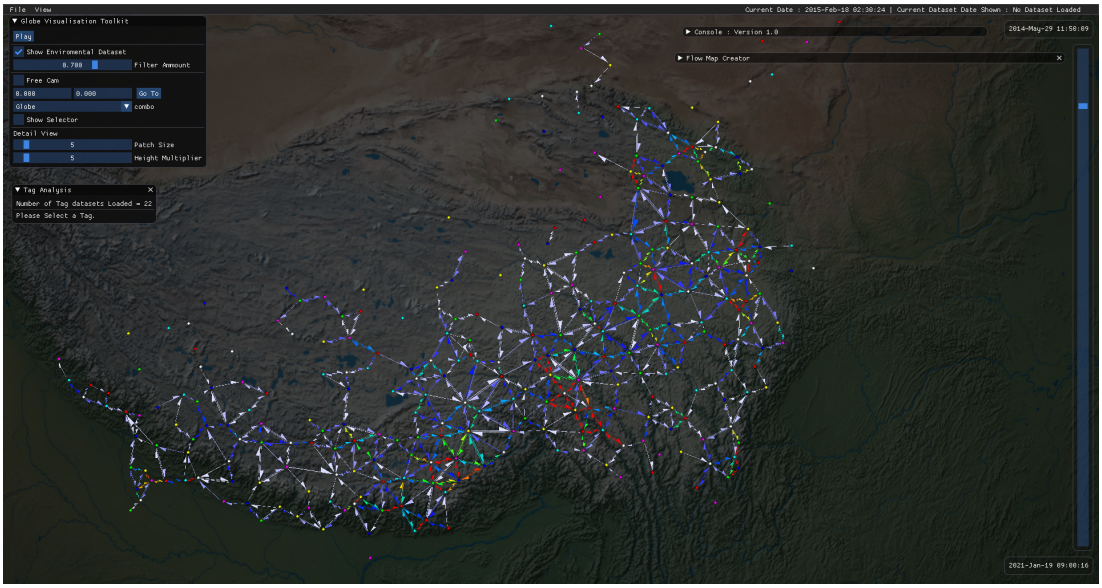


Figure 5.4: GPGPU acceleration enables for large scale movement datasets to be processed. Here a dataset by Sherub et al [SBWW16] that contains over six million datapoints can be processed in three seconds. The resulting flow map is then displayed in global view.

by first transforming each relocation to the distance moved for that relocation then performing segmented reduction on the result. The annotated visits array is used to then count the number of flow transitions that occur.

To store the flow counts, the number of transitions between each centroid, a N_c*N_c matrix is created, with N_c being the number of centroids. This allows for counting of flow transitions between centroids in both directions. Stream compaction is performed on the annotated visits array, with relocations where the next point contains a different centroid being flagged as a flow transition. The index of this transition within the flow count matrix is equal to $c_i * N_c + c_j$, where c_i is the current closest centroid and c_j is the closest centroid of the next point. The flow count matrix 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 much smaller than the number

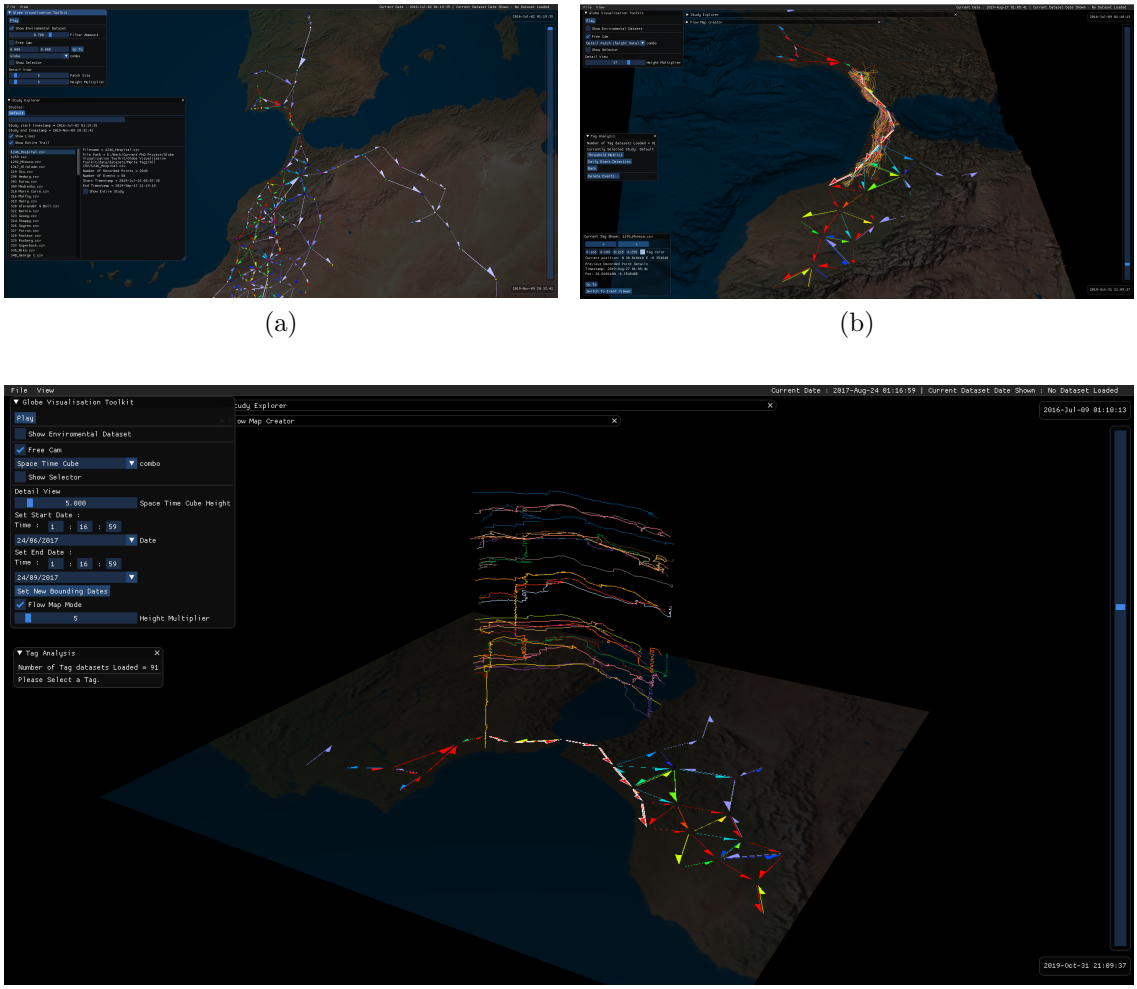
of relocations. The compacted array is then used to annotate the trajectory with the approaching flow using a scan operation. This annotates the trajectory with the index of the approaching flow, with the final relocations assumed to be part of the previous flow transition. This is shown within Figure 5.3. With these steps complete for all trajectories, the visualisation can now be built.

5.3 Flow Map Visualisation

Using the flow count matrix, centroids of characteristic points and trajectory data annotated with the approaching flow three distinct visualisation methods can be constructed: On the surface of the interactive globe, on a detailed patch of height data and a spatially filtered space time cube. Each method is interchangeable, with the visualisation constructed as necessary to allow for minimum resources used and interactive switching between different views.

5.3.1 Global view

The first method of visualisation is the global view. This view allows for global patterns to be visualised. It is the initial view upon creating a flow map, allowing the user to investigate the flow map as a whole, allowing for an overview of the level of generalisation so that adjustments can be made. It is useful for identification of migratory corridors, common paths that a migratory species travel, and excels in the visualisation of flow maps that have a high level of generalisation. To create the points on the globe each centroid is converted to ECEF coordinates using Equations 4.3.1 - 4.3.3 in Chapter 4. The flow count matrix is then checked for any values above zero, creating the geometry needed between the appropriate centroids.



(a) (b)

(c)

Figure 5.5: The visualisation of a flow map calculated from white stork migration data. Initially calculated within global view (a), then within the detail view with the ETOP01 digital elevation model (b) and finally using the flow map to create a spatially filtered space time cube (c).

5.3.2 Detail view

Digital elevation models are commonly viewed with environmental datasets to assist in determining the reasoning behind movement patterns. With the globe being rendered as a 3D model applying a digital elevation model to the geometry is possible, however maintaining a real-time framerate can become unfeasible for large Digital Elevation Models. For example the ETOP01 Digital elevation model contains over 233 million points [Ama09]. To facilitate the visualisation of smaller scale movements, as well as investigation of low maps with lower levels of generalisation a detail view is implemented. This detail view allows for visualisation of height data over a smaller area, with the focus to be moved across the globe as desired, with geometry updating in real time.

During inspection of an environmental dataset, tag or flow map a centroid may be selected by the user, this point may also be selected automatically during tag playback to follow the tag. A patch size is then determined by the user and the toolkit will select the appropriate boundaries around the selected point. The appropriate digital elevation model values and a desired height multiplier (used to exaggerate height features) will then be sent to the GPU for processing. A kernel is then invoked that creates the geometry required for the detail view.

With the height data plotted and normal calculated the flow map can then be placed over the detail view. Any centroids that lie outside of the current view are filtered out and any flows connecting to invalid centroids are not shown. When the initial detail view is created the user can move it across the globe, either by switching back to the global view or using the arrow keys to move or following a tag that is playing. An advantage of using CUDA to render the environmental variables to a texture is that the detail view can also visualise the environmental variables from Chapter 3 without needing to recalculate values. With only texture coordinates calculated

within the geometry kernel the detail view can also visualise the environmental data at a smaller scale. This means that during playback the detail patch will be updated continuously to allow for the real-time visualisation of both a digital elevation model and environmental datasets surrounding a tracked tag.

5.3.3 Spatially Filtered Space Time Cube

With the global view allowing for the visualisation of large scale migratory patterns and the detail view allowing for the investigation of low levels of generalisation and small scale movements the space time cube allows for the investigation of not only where relocations take place, but also when. Space time cubes have a long and detailed literature that is covered within Chapter 2.5, but a brief overview is given here. Initially hand drawn by Hägerstraand [H$^+$68] space time cubes allow for the visualisation of spatiotemporal data. The position of each relocation is plotted normally, in our case the longitude and latitude of each relocation, but with the height of each relocation being dictated by the time in which the trajectory reached that point, with higher points occurring after relocations below. While this method has advantages in being able to quickly identify patterns such as stopovers quickly, it has been found that they can become difficult to use as the number of points increase. Similar to the use of density maps to aggregate map data [DV10a, BND$^+$20] space time densities can alleviate the issues of large amounts of data. This does however obfuscate the data somewhat, with direction of each of the trajectories being lost. Recent developments, such as the time mask by Andrienko et al. [AAC$^+$17a], have looked at filtering space time cubes to only display meaningful data. This level of control allows for the interactive investigation of multiple processes.

To allow for the interactive adjustment of generalisation, as well as the preservation of direction while reducing the amount of visual clutter for large datasets, we

apply a similar filtering technique to space time cubes, using the flow map calculated to filter for direction and position in space. First, a space time cube is created that can then be filtered. Instead of passing the start and end dates of the space time cube to the shader and performing the calculations for the height of each vertex within the vertex shader, a CUDA kernel is invoked. This is done to avoid the unnecessary recalculation of the trajectories within the space time cube. While this may have little effect on smaller datasets, as the trajectories increase in both number and size the, frame rate will decrease when a shader is used to compute height values. The use of CUDA allows for the advantages of GPU acceleration without recalculation every frame, only updating the space time cube when desired. The shader can then discard any vertices that do not lie within the specified bounds of the space time cube. Further filtering can then be performed, using the annotation of each trajectory with the approaching flow. Via the use of a shader storage buffer object, the index of each selected flow is sent to the shader. If the relocation within the trajectory is not approaching a selected flow the shader discards that element, leaving only the unfiltered trajectory that is desired. This can be seen in Figure 5.5 (c).

5.4 Results

All testing was performed using a system running Windows 10 with an Intel i7-7700K CPU @ 4.2GHz, 16Gb RAM and NVIDIA GTX 1080. The dataset used for testing is the migratory white stork dataset from Acácio el al. [FCRA19] and was downloaded directly from Movebank [Mro18].

The clustering step has seen an increase in performance due to GPU acceleration. Andrienko states 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.

The adjustment of the tight and loose distances allows for a number of varied flow maps. As the value of the tight distance decreases the number of centroids increases, this in turn lowers the level of generalisation. The adjustment of the T1 allows for the adjustment of centroids based on nearby points, rather than just the points making up the centroid. As the difference between the T1 and T2 increases the effect of other clusters on centroid calculation also increases. As the value of T1 approaches the value of T2 a more regular set of centroids with even spacing is created, making the result more similar to the clustering method by Andrienko and Andrienko.

It should be noted that with this method the worst case has a time complexity of $O(n^2)$. This occurs when each tag event makes up a cluster containing only the single tag event, resulting in the partition step being performed n times with a time complexity of $O(n)$. This is in contrast to the worst case of the clustering method by Andrienko and Andrienko. In their case, the worst case would involve every tag event being within a single cluster. This results in a computational complexity that approaches $O(n^2)$ as the number of events within each centroids increases.

As the radius parameter, and thus the amount of generalisation, is increased, the number of points within each centroid also increases, thus resulting in longer computation times for the CPU sequential method. On the other hand, the computational complexity, and thus the computation time, is reduced in this case for our GPU method. We argue that this is more intuitive when investigating large amounts of movement data, with higher levels of generalisation requiring less computation time than visualisations with higher levels of detail.

N	GPU Annotation Time (ms)	CPU Annotation Time (ms)	Speedup
359	6.89	4.69	x0.6
914	8.24	12.06	x1.5
1204	7.82	15.91	x2
2190	8.03	30.16	x4
4410	7.96	65.79	x8
5675	7.81	74.50	x9
7811	8.47	108.91	x13
35311	21.02	371.09	x18
63116	79.95	1148.94	x14
83514	99.99	1592.88	x16
110632	116.48	2020.73	x17
262248	175.47	3240.21	x18
353997	331.86	6526.05	x19
414493	396.90	7620.22	x19
554788	609.65	11257.8	x18
617222	561.62	11335.8	x20
869502	769.67	15995.2	x20
927100	829.96	17067.5	x21

Table 5.1: Time required to segment and annotate a trajectory of a white stork with the closest centroid and to calculate visit descriptors, such as time duration and movement distance.

As can be seen from Table 5.1 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.

As can be seen from Tables 5.2 and 5.3 the detail view and spatially filtered space

N	Detail View Update Time (ms)
360000	2.13
518400	2.43
705600	3.08
921600	3.73
1166400	4.53
1440000	5.59
1742400	6.27
2073600	7.25
2433600	8.29
2822400	9.65
3240000	10.90
3686400	12.01
9000000	28.05
12960000	40.24

Table 5.2: Time required to update the detail view mesh with data points from the ETOP01 digital elevation model.

time cube view can rendered in real time. For the detail view, selection from the ETOP01 digital elevation model that contains over 230 million data points can occur in real time. A detail view containing 360,000 points can be created and updated in 2 milliseconds. This allows for the detail view to update during playback, following a desired tag. The user can also playback multiple tags and adjust the position of the detail view as desired. Space time cube creation has also benefited greatly from GPU acceleration. To update the trajectory containing just under one million data points only requires 4 milliseconds. This allows for the creation of space time cubes containing large amounts of data. This alone would not be of use to most researchers due to the abundance of data causing the visualisation to become difficult to interpret. With the use of spatially filtered space time cubes these visualisations become usable, with variable levels of generalisation available. This is shown within Figures 5.6 - 5.8 With a set of trajectories containing 7.5 million relocations and a flow map containing 57 centroids, the space time can be created in 85 milliseconds

N	Time (ms)
2071	0.66
30088	0.67
81621	0.88
89428	0.96
97653	1.37
143135	1.19
189057	1.29
225867	1.38
252465	1.41
273303	1.49
355933	1.98
366262	2.02
599915	2.69
714586	3.25
831513	3.70
906160	4.57

Table 5.3: The time required to create the visualisation of a trajectory on a space time cube. This time includes the transformation of the geodetic coordinates, calculation of trajectory height using each relocations timestamp and filtering of data that lies outside of the current visualisation.

and maintain a framerate of 300fps. This allows for the interactive adjustment of the position and time thresholds of the space time cube.

With all of these optimisations combined a user can create a flow map with 300,000 event points, clustering to 318 centroids and generalise 20 trajectories totalling 7.5 million relocations in just over one second. This flow map can then be visualised in real time using three methods, a global view ideal for large scale and high generalisation, a detail view for visualisation of digital elevation models and a low generalisation and finally spatially filtered space time cube, allowing for users to visualise not only where trajectories interact but when while avoiding the pitfalls of visualising large amounts of data.

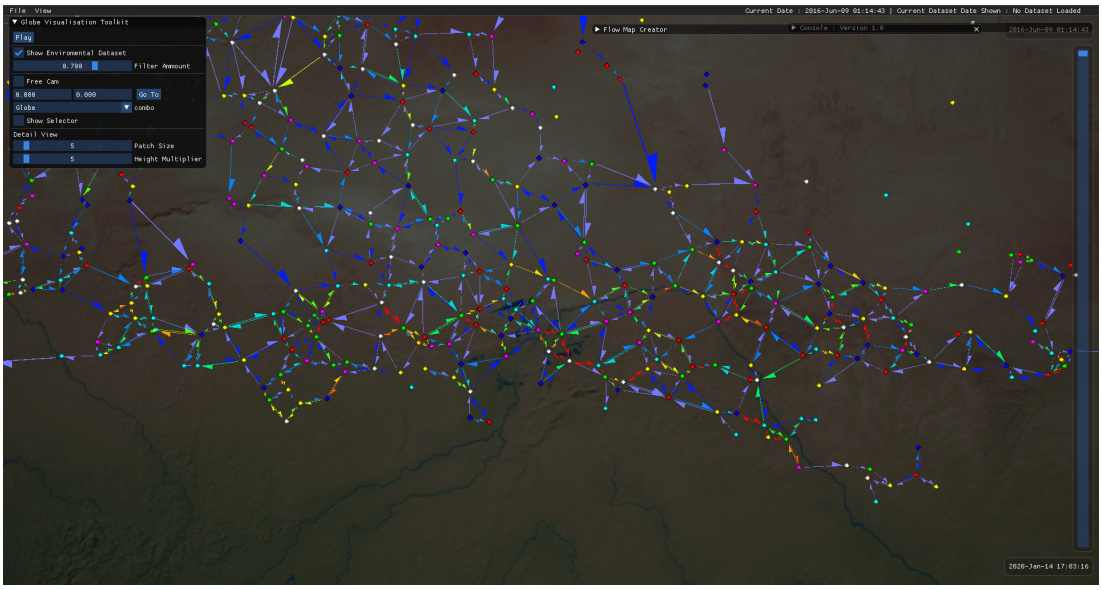


Figure 5.6: The flow map generalisation of a white stork dataset, with a cluster size of 60Km.

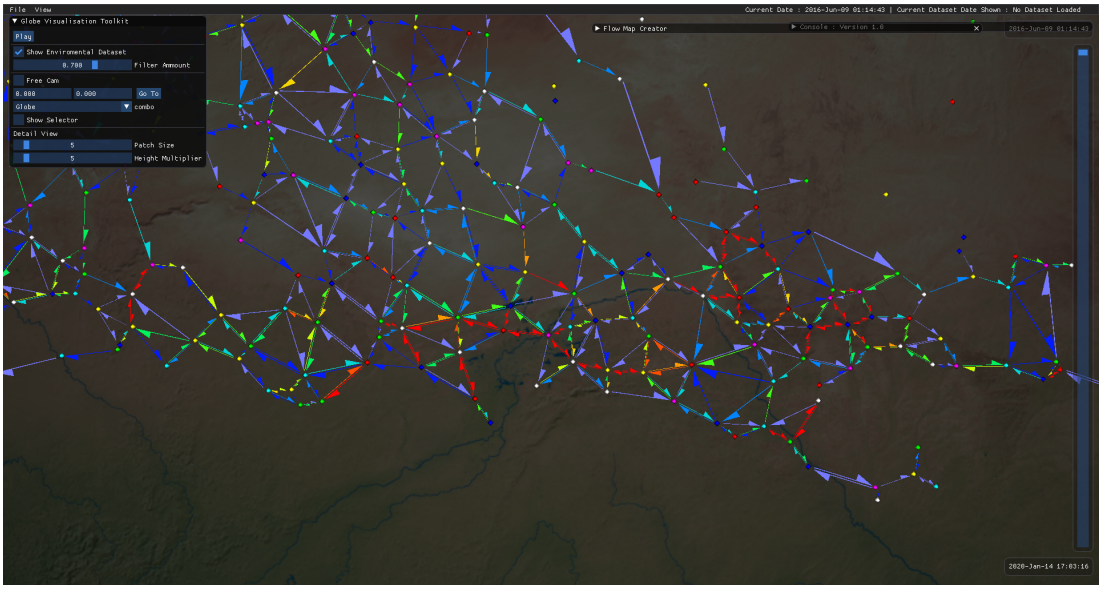


Figure 5.7: The flow map generalisation of a white stork dataset, with a cluster size of 80Km.

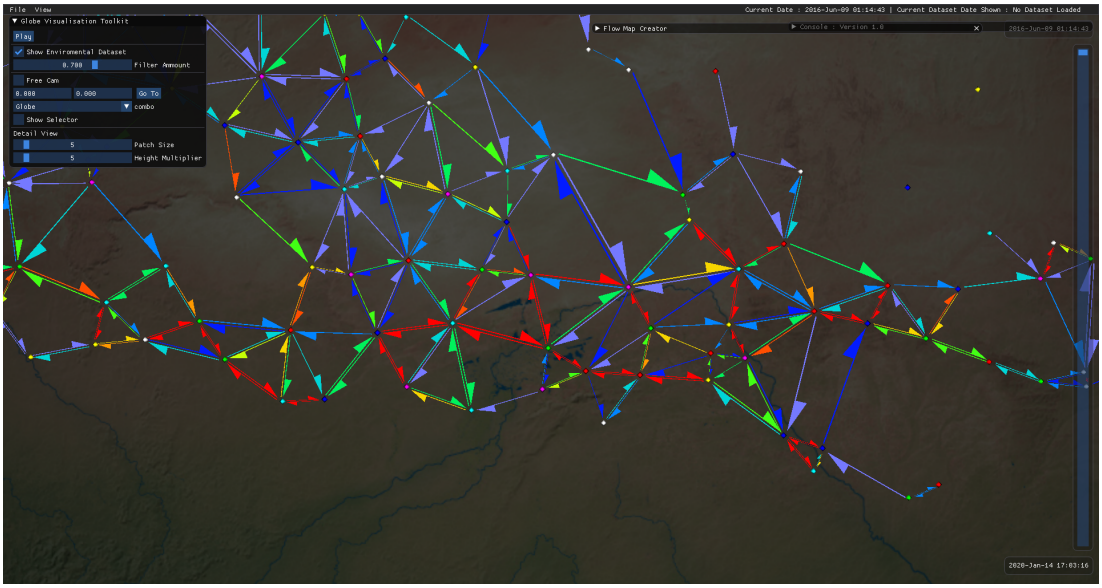


Figure 5.8: The flow map generalisation of a white stork dataset, with a cluster size of 120Km.

5.5 Conclusion

With both tag event detection of characteristic points and level of generalisation significantly effected by their parameters, any resultant flow map calculated from the data will have significantly diverse results. With long waiting times between each flow map there can be a disconnect between parameter adjustment and the resultant visualisation. Here we have presented a GPGPU accelerated clustering and segmentation method for the creation of flow maps that allow for the interactive generalisation of large movement datasets.

The resultant flow maps can then be visualised in a number of ways: A global visualisation allowing for the investigation of migratory studies that cover large distances, a detailed view that allows for the generalisation of smaller scale movement visualised along side the ETOP01 digital elevation model, and a filtered space time cube visualisation, a novel method that allows for the visualisation of trajectories within a space time cube with real time adjustment of the level of generalisation.

As noted by Gorodov et al [GG13], there are five common issues when attempting to visualise a large amount of data. By utilising GPGPU acceleration, the high performance cost is reduced, with millions of datapoints able to be processed interactively. Through the use of aggregate flow maps, rather than just density surfaces, the obfuscation of data is reduced, with directional information still available. The filtering of a space time cube also allows for the reduced obfuscation of data within areas of interest, with the selected parts of the flow map showing the underlying trajectories. This filtering of data, alongside the flow map, vastly reduces the amount of visual noise when investigating large scale movement studies.

Chapter 6

R Package Integration and Case Studies

6.1 Introduction

Within this thesis we have presented a number of GPGPU accelerated techniques for visualising and analysing large amounts of movement data. While these methods may be computationally efficient, the development of these methods is of no use to the wider field without reproducible results that are accessible to movement ecologists. To this end, throughout development, communication between researchers of differing disciplines has been at the forefront. This interdisciplinary nature has driven the research throughout this thesis from the beginning. The development of the Global Animal Movement Toolkit (GAMT) has been an iterative process over the last 4 years, with development influenced by not only the technical capabilities of GPGPU technology but current visualisation and analytical techniques useful to movement ecologists.

Within this chapter we present the toolkit in its entirety. Developed with C++ and OpenGL, it can load both environmental and movement data, allowing researchers to use the techniques described in Chapter 3 to visualise their movement data with ecological context. We show that this movement data can be visualised

alongside its movement metrics calculated using the methods described in Chapter 4 in real time, allowing for interactive graphs to be used to investigate the effect that the parameters have on the results. GPGPU acceleration also allows for the toolkit to seamlessly switch between different visualisation methods of the same data. For example, between the aggregated flow diagrams of Chapter 5, spatially filtered space time cubes and the global environmental visualisation of Chapter 1.

R is an interpreted programming language that is commonly used in movement ecology and other statistical fields [Xie13, SFA19, Sou11, MS18, HRBP15]. Due to R being an interpreted language, it does not lend itself easily to the techniques throughout this thesis. Its use throughout movement ecology and current research workflows, alongside the ease of integration of packages from other researchers, merits integration with the techniques within this thesis. To this end we will also show the R package that has been developed to enable download of commonly used global environmental datasets, then transfer them to GAMT ready for visualisation. The R package also handles the transfer of movement data from a standard R dataframe to a set of CSV files ready for visualisation. Finally, we show the R package and toolkit being used within a case study with Jethro Gauld.

6.2 The Global Animal Movement Toolkit (GAMT)

The global animal movement toolkit is a C++ based desktop application for use by researchers to investigate and visualise large scale animal movement data. Using the OpenGL interoperability with CUDA, the toolkit provides a pre-built heterogeneous computing environment for NVIDIA GPUs. The toolkit allows for all analysis from Chapter 4 to be performed without the need for the user to explicitly handle the data transfer from host to device. The toolkit handles these transfers via the CUDA-Controller class. This class acts as a mediator between the CPU and the GPU, with

the CUDA compiler only compiling functions that lie within the CUDAController and any included headers. This allows for an expandable heterogeneous computing environment, with the only requirements for a new function being its inclusion in the CUDAController class and an appropriate UI element added. The ease of expandability allows for new methods to be implemented with relative ease, with each new metric calculation or analysis technique able to be implemented in a separate development environment and added to the tool when ready. Due to the toolkit also processing the visualisation of the trajectory data, the CUDAController is also able to utilise the OpenGL interoperability with CUDA to optimise the transfer of data between the toolkit and CUDA kernels. With the geodetic coordinates used for visualisation, contained within a vertex buffer object, metric calculations can be performed using this data directly, rather than including another redundant memory transfer.

Due to being initially designed to handle Movebank datasets, the toolkit reads and exports comma separated values (CSV) files. CSV files, however, are loosely defined, with each study defining their headers in differing orders with differing names. To circumvent this, the toolkit asks for user input upon loading a group of CSV files. This UI allows for users to define the contents of their data. It should be noted however that there are four required headers: longitude, latitude, altitude and timestamp. These basic headers are used to create the different visualisations. These columns are also used to filter out erroneous data and missing values when loaded. As each CSV is loaded, if a longitude or latitude value is missing it is removed from the dataset. If an altitude value is missing it is instead replaced with a zero value. Whenever either of these cases is found it is reported to the user after the file is loaded. To allow for more flexibility when loading multiple trajectories, the toolkit treats each CSV file within a folder as a single trajectory. This is done to allow for selective loading of

Figure 6.1: The Global Animal Movement Toolkit allows for the real time calculation and visualisation of movement metrics alongside raw trajectory data. Here, the net squared displacement of a migrating white stork from 2019, has been calculated while the trajectory (cyan line on globe) is visualised. The current position of time is highlighted on the graph in red.

trajectories within a study. When a study is compiled into a single CSV file it can be separated via the use of the R package shown later in this chapter.

When exploring large amounts of data, filtering out data not currently under investigation can be useful to reduce the amount of visual noise within the visualisation. Here, the study explorer has been implemented to allow for the filtering of data. Upon loading, each trajectory is placed within the default study. A study is defined as a set of trajectories that are to be processed and visualised together. The user can then use the study explorer UI to selectively group trajectories into separate user defined studies, with the trajectories that make up a study being visualised and processed together. At any point the user can select the default study to process and visualise all trajectories if desired. This level of interaction and control over which trajectories are being processed is applied to all of the previous methods within this thesis. Here we see the advantage of rapid processing of datasets enabled by the use

of GPGPU computing, the user can interactively change between studies and any visualisation is updated in real time to show the new study, with any analysis only being performed on the selected study, saving memory transfer time. If desired, a user can also visualise and analyse a single trajectory at a time within a study.

With movement metrics being calculated for entire studies, it may be useful to have an interactive graphing environment to visualise the different metrics together, or the same metric over an entire study. To this end the toolkit utilises the ImGui library alongside the ImPlot plotting library to create a user interface that allows for users to graph their data [Cor]. As shown in Figure 6.1 the user can visualise their trajectory data alongside any movement metrics calculated, with the current timestamp of the visualisation being shown on the graph as well. To assist in visualising movement metric thresholds for event detection, this too can be shown on the graph.

A key contribution of this toolkit is its ability to change visualisation methods and entire environmental and movement datasets in real time. Enabled by the GPGPU accelerated methods described in earlier chapters, the user can interactively investigate their data via different methods and switch between them with minimal delay. For example, a user can calculate a set movement metrics for their data and visualise it on the globe. They can then change to the space time cube view to investigate if movements are occurring at a similar time, with the environmental data still being visualised. With too much data being visualised the space time cube becomes too noisy, so a separate study can be used to filter out unwanted data. This can then be further refined by calculating a flow map for the remaining data that can then be adjusted for differing levels of generalisation. The user can then select individual flows in which the user intends to investigate. In total this can be performed on a movement study containing millions of data points interactively, and at any point the user can perform analysis on the trajectories, such as step speed and identifying

migratory segments, and graph their movement metrics alongside the visualisation.

6.3 R Package

While the Global Animal Movement toolkit provides a number of useful visualisation and analytical methods, its bespoke nature as a C++ application leaves it isolated from other techniques implemented within the field of movement ecology. Joo et al. have investigated these packages in detail and have seen that many of these R packages have also been developed in isolation [JBC+19]. In order for users to maximise their workflow in an accessible manner, we have developed an R package that allows for communication between the R session and the Global Animal Movement Toolkit. This allows for the user to utilise the toolkit within their workflows alongside other packages.

While there are now robust wrapper R packages that support the creation of visualisations using OpenGL [ANZ03] and the development of CUDA kernels in R [DPS+10], creating an optimised toolkit that can utilise the OpenGL CUDA interoperability without dedicated IDE support would be impractical. Due to this, the toolkit was implemented as an isolated application that communicates with an R session via the use of an intermediate file. This allowed for dedicated debugging tools to be used during development of GPGPU accelerated methods, while still allowing for the toolkit itself to be a part of the ever growing library of R packages used by movement ecologists. To implement this, the cereal library was used to create a JavaScript Object Notation (JSON) file. This file contains each instruction that is called by the R session that requires an output from the C++ application. The JSON file is structured such that it contains a list of each of these instructions, with each instruction containing any information needed for it to be completed. To begin, the user first specifies the installation directory of the toolkit so that the R session can

find the relevant files and open the JSON instructions file. JSON was used for its readability so that if there are any errors in the construction of an instruction the user can open the instruction file directly to see the error and correct their parameters. With the installation located and the instruction file opened the R session can begin sending instructions to the toolkit. When running, the toolkit periodically checks for the last write time of the JSON file and if the file has been updated since the last read the toolkit will follow the last instruction in the file.

6.3.1 Dataframe Loading

One such instruction is the separating of a single R data frame that contains a set of trajectories to be visualised into separate CSV files for visualisation. In order for the toolkit to load a set of trajectories, each must first be separated into individual CSV files that each contain a single trajectory. This is done to minimise the amount of data loading on subsequent uses if only a partial subset of the data is desired. This also allows for studies to be pre-defined by the file structure of the CSV files. Within the import folder if a set of CSV files are within a sub folder they are automatically placed within a study. The only requirement being that all CSV files within the import folder must have the same format, as the toolkit assumes that the format defined by the user applies to all CSV files. To separate an R data frame into separate CSV files, the R package first requires the user to specify a column within the dataframe that uniquely identifies each trajectory. This is then used to separate the data frame using the split function in R. Each of the data frames are then written to a CSV filed named after the unique identifier within the specified column. Any other columns within the data frame are also saved alongside each trajectory, allowing for the user to perform a step-wise function in R and export it to the toolkit if desired.

6.3.2 Environmental Dataset Loading

The other main function of the R package is the automation of downloading and loading large amounts of environmental data into the toolkit for visualisation. As stated in Chapter 3, the toolkit utilises the Global Data Abstraction Library (GDAL) to support a multitude of differing global data formats. While this allows for a certain amount of automation when loading environmental datasets, the slight changes in formats and naming conventions between datasets requires a level of user input to ensure a correct visualisation is achieved. This can significantly delay the loading of large amounts of data that are split across a multitude of files. To speed up the input of large amounts of data, the environmental file list structure is used. To create a file list, each GDAL compatible file directory (HDF4, HDF5 NETCDF, etc) that makes up each desired timestamp is placed within a JSON list with its desired timestamp. Both the list of filenames and timestamps are passed to the envDataLoad function as parameters. Any other associated data are also added to this instruction, such as data type, pixel registration, dataset name and the time difference between timestamps if multiple timestamps are within the same file. This JSON save file is then placed within the same directory as the environmental dataset files. An instruction pointing to this file can then be created to inform the toolkit that a set of files are to be loaded. This file structure allows for researchers to create a single file that can then be used repeatedly to load the same datasets with the same timestamps for each environmental dataset between sessions.

When investigating movement within an ecological context there are common environmental factors that are considered. These include temperature, vegetation and wind [MS19]. This has given rise to global dataset repositories for environmental data, such as the Copernicus Data Store (CDS) [MS19]. To simplify the process of obtaining and visualising these datasets, the GAMT package uses the ecmwfr package

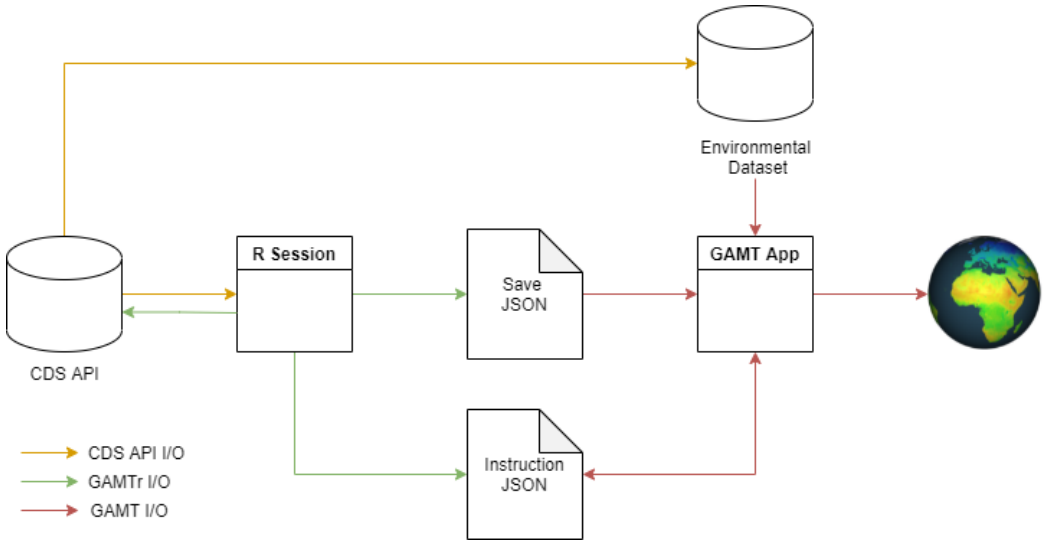


Figure 6.2: Systematic outline of the environmental dataset loading within GAMTr. The user can use GAMTr to access the CDS API to request commonly used environmental variables, such as temperature and vegetation indices. This will then download the requested dataset and create a save JSON file that contains each of the separate files used. The instruction JSON is then updated, GAMT will then automate the loading of all desired environmental data to the output the visualisation using the methods from Chapter 3.

[HSC19] and the CDS API to automate the download of common environmental datasets. First, the user logs into their CDS account via the usual ecmwfr function calls. Next, the user designates which environmental dataset is desired via any of the showERA5X functions. The time range is then defined, and a sequence of calls to the CDS API are then performed to download all the data within the designated time range, with one file for each month. If the file has been previously downloaded, then the file is skipped to save downloading time. Once all the environmental data files are downloaded the R package then creates a JSON environmental list, described earlier, and then sends an instruction to the toolkit to load the environmental data. This implementation of the environmental dataset list allows for users to define a set of compatible user created environmental files for quick visualisation within the toolkit. The GPGPU accelerated visualisation methods outlined in Chapter 3 can then be

used to switch between these datasets, or timestamps of the same dataset, in real time. For users just beginning to visualise their data, the CDS API wrapper allows for commonly visualised environmental datasets to be easily downloaded and sent to the toolkit for visualisation.

Once the user has completed their visualisation and analysis of their data within the toolkit, the user can utilise the annotation method outlined within Chapter 4 to output their data, with any movement metrics appended, into an R dataframe. First, the user selects which data they want to export using the study explorer. Once the annotation is complete the R session can read the export folder within the installation directory and load each into an R data frame. With the data back into the same format as the data frame initially used the user can then perform any further analysis on the entire study within their R session.

6.4 Case Study: Using The Global Animal Movement Toolkit To Assist With Predictive Modelling Of Bird Sensitivity To Wind Farms And Power Lines.

Co-author Jethro Gauld.

As we transition to a zero carbon electricity grid by rapidly expanding renewable energy generation capacity, there is a danger that we may neglect the biodiversity crisis in the name of solving the climate crisis [KBMK$^+$19, RCS$^+$19]. This will also require significant investments in the electricity transmission and distribution network. One way in which poor spatial planning of wind turbines and power lines can negatively affect bird populations is by increasing the risk of mortality from collision [KBMK$^+$19].

It is most optimal to consider potential wildlife impacts at the site selection and

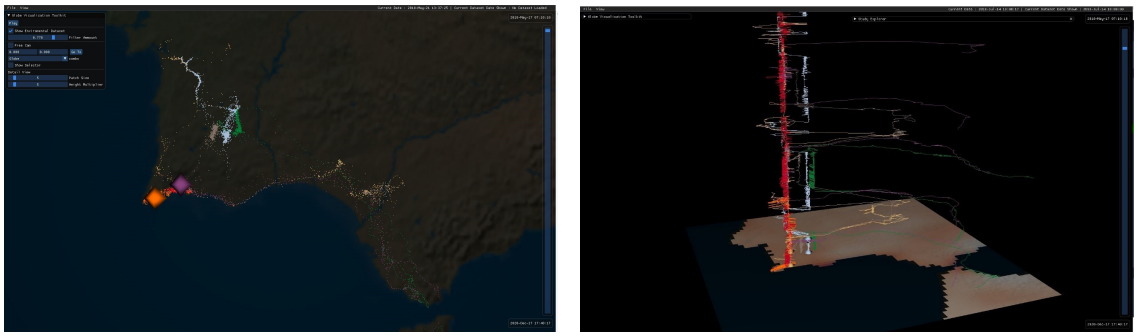
scoping stages of new wind farm developments as this can help avoid additional costs related to mitigation later in the development pipeline and public perception of the project [KBMK+19]. Analysis of GPS movement data of birds to produce sensitivity maps can assist with this by helping us understand where and when birds are most at risk from new developments at the flyway or landscape scale [BBB+18, TRK+18]. Sensitivity maps are therefore a potentially important tool to aid final site selection and site-based surveys to inform the layout of the development and any specific mitigation requirements to reduce risks to birds.

A limitation to this approach is the availability of tracking data [GSA+ng]. While costs of GPS loggers are becoming more economical, other costs associated with deploying loggers such as labour, data costs, training and time are likely to remain significant barriers [HHH+19] resulting in gaps in the available trajectories where sensitivity cannot yet be assessed by directly analysing the GPS tracking data. Here we demonstrate a method for how we can fill these gaps in the data using simple predictive modelling. Firstly we seek to understand the key environmental predictors of flight height in white storks Ciconia ciconia using tracking data from Portugal. We then use this to produce a raster surface with predicted probabilities of birds being present at danger height or not in a given grid cell and visualise this using the Global Animal Movement Toolkit (GAMT) and methods developed within this thesis.

6.4.1 Methods

Data Search

We used high resolution GPS telemetry data for white stork Ciconia ciconia sourced from the White Stork Adults 2018 data set via Movebank [WM19], which provides location and altitude of the birds (Movebank, 2019). This soaring species is a useful subject for this analysis because it is prone to colliding with wind farms and power



(a) (b)

Figure 6.3: Visualisation of raw bird movements within Iberia using GAMT. (a) The raw trajectory data highlights that without a predictive model we would only be able to assess sensitivity for regions where we have tracking data. The visualisation used is the global view with each relocation marked with a single point. (b) Visualisation of bird movements in space and time using a 3D space-time cube with wind speed in GAMT. The space time cube allows for the timings of migrations to be compared.

lines [MSH+20, DMÁM+19] and the known distribution of this species in Iberia is well represented by the available GPS tracking studies. As summarised within Table 6.1, a number of environmental variables were used to annotate the trajectory data using the annotation method described within Chapter 4. These include land cover, orthographic uplift, tree cover, distance from the coast , NDVI, and weather variables. Height of the land surface was derived from the 30m resolution, 5m vertical accuracy STRM-GL1 digital surface model (DSM) available to download in height above average sea level (ASL) and height above ellipsoid (HAE) from the Opentopography Portal [NN00, Nat19]. We use data from [DSTE20] and the Open Infrastructure project (Garret, 2018) to create the infrastructure density layer.

Data Handling

Bird tracking data was prepared for analysis in R using the Tidyverse package to check for duplicates, filter out lower precision GPS locations and remove GPS locations outside the study area [HFHM15]. Terrain data (roughness index (TRI), aspect and slope) was processed using the Raster Package [HvE12] and then GAMT was

Dataset	Resolution	Origin	Resolution	Appended in GAMT	Appended in R	Appended in Movebank ENV Tools	Comments
Wind_U	0.05°	ERA5 from the Copernicus Data Store	0.05°	Y	N	N (Possile)	
Wind_V	0.05°	ERA5 from the Copernicus Data Store	0.05°	Y	N	N (Possile)	
Surface Temperature	0.05°	ERA5 from the Copernicus Data Store	0.05°	Y	N	N (Possile)	
Air Pressure	0.05°	ERA5 from the Copernicus Data Store	0.05°	Y	N	N (Possile)	
Orographic Uplift	0.05°	Movebank ENV tools	0.05°	N	N	Y	A Movebank product but going forward we will create our own Upflift surfaces.
Thermal Uplift	0.05°	Movebank ENV tools	0.05°	N	N	Y	A Movebank product but going forward we will create our own Upflift surfaces.
Landcover LCCS1	500m	MODIS	500m	N (Possible)	N (Possible)	Y	For large data sets ENV tools appeared to struggle. In future would be faster to append in GAMT.
Land Use LCCS2	500m	MODIS	500m	N (Possible)	N (Possible)	Y	For large data sets ENV tools appeared to struggle. In future would be faster to append in GAMT.
TRI (Terrain Roughness Index)	90m	STRM-GL1 30m DEM	90m	Y	N (Possible)	N	TRI was calculated using the Raster Package in R from the STRM-GL1 30m DEM
NDVI 1km Resolution	0.01°	MODIS	0.01°	N	N	Y	ENV Tools seems to struggle, GAMT is much quicker.
NDVI 0.05 Decimal Degrees	0.05°	MODIS	0.05°	Y	N	N	
Percent Tree Cover	250m	MODIS Land VCF 250m Yearly Terra Percent	250m	Y	N (Possible)	N (Possible)	Possible to append by all three methods
Elevation Relative to Elispoid (m)	30m	STRM-GL1 30m DEM	30m	Y	Y	N	Possible to append in GAMT and R but GAMT is Faster
Elevation relative to Mean Sea Level (m)	30m	STRM-GL1-Ellipsoidal 30m DEM	30m	Y	N (Possile)	N (Possile)	Possible to append in GAMT and R but GAMT is Faster
Slope	30m	STRM-GL1 30m DEM	30m	Y	N (Possile)	N (Possile)	Slope was calculated using the Raster Package in R from the STRM-GL1 30m DEM
Aspect	30m	STRM-GL1 30m DEM	30m	Y	N (Possile)	N (Possile)	Aspect was calculated using the Raster Package in R from the STRM-GL1 30m DEM
Distance from a River	0.01°	World Bank Major Rivers	0.01°	Y	N (Possile)	N	Calculated in R from the Major Rivers data set with the Raster Package to produce a distance raster then appended in GAMT
Distance from a Road	0.01°	Global Roads Open Access Data Set (gROADS), v1 (1980–2010)	0.01°	Y	N (Possile)	N	Calculated in R with the Raster Package to produce a distance raster then appended in GAMT
Distance from Transmission Power Line	0.01°	Open Infrastructure Project	0.01°	Y	N (Possile)	N	Calculated in R with the Raster Package to produce a distance raster then appended in GAMT
Distance from Wind Turbine	0.01°	Dunnet et al. 2020	0.01°	Y	N (Possile)	N	Calculated in R with the Raster Package to produce a distance raster then appended in GAMT
Distance from the Coast	0.04°	NASA Ocean Biology Processing Group	0.04°	Y	N (Possile)	N (Possile)	Calculated in R with the Raster Package to produce a distance raster then appended in GAMT

Table 6.1: Summary of data sets used in the analysis and how they were appended.

used to visualise the result and annotate the trajectory data. Infrastructure data was processed in QGIS ; this was used to create a proximity raster allowing distance from wind turbines, roads, rivers and power lines to be appended to each data point and two rasters containing the density of wind turbines or power lines in each pixel (5x5km). Displacement between GPS locations was calculated both using the Geosphere package [HWV15] and the GPGPU accelerated method within Chapter 4. This was done as to compare the outputs. Speed was calculated for each fix per individual bird by dividing the distance between GPS locations in metres by the time in seconds to yield a speed estimate in m/s. GPS locations were then filtered to a minimum 5 minute fix interval before further analysis to ensure independence of the GPS locations included in the analysis. We calculate height above ground by subtracting the height of the ground from the altitude for each GPS location. Where this is in HAE, we use a DSM in that reference. Where the altitude is measured relative to ASL we use a DSM in ASL [PFD+17]. Each GPS location was then categorised as breeding season, migratory or non-breeding season using the migration detection methods outlined in Chapter 4.

Defining Height

There are numerous methods used to determine whether a bird is in flight, or not. These generally rely on some measure of the bird's behaviour such as flight height or speed at the time a GPS fix is recorded. In defining a fix as in flight we need to acknowledge the potential errors associated with measuring speed and altitude above the ground [PDH+18]. These errors are related to the number and position of satellites used by the GPS device to measure altitude this vertical error given by many GPS devices is typically in the region of 1.5m but can be as large as 31m [MSH+20]. There is additional error arising from the DSM (±5m) used to determine the height of the ground surface [PCD+20]. This can result in erroneously negative

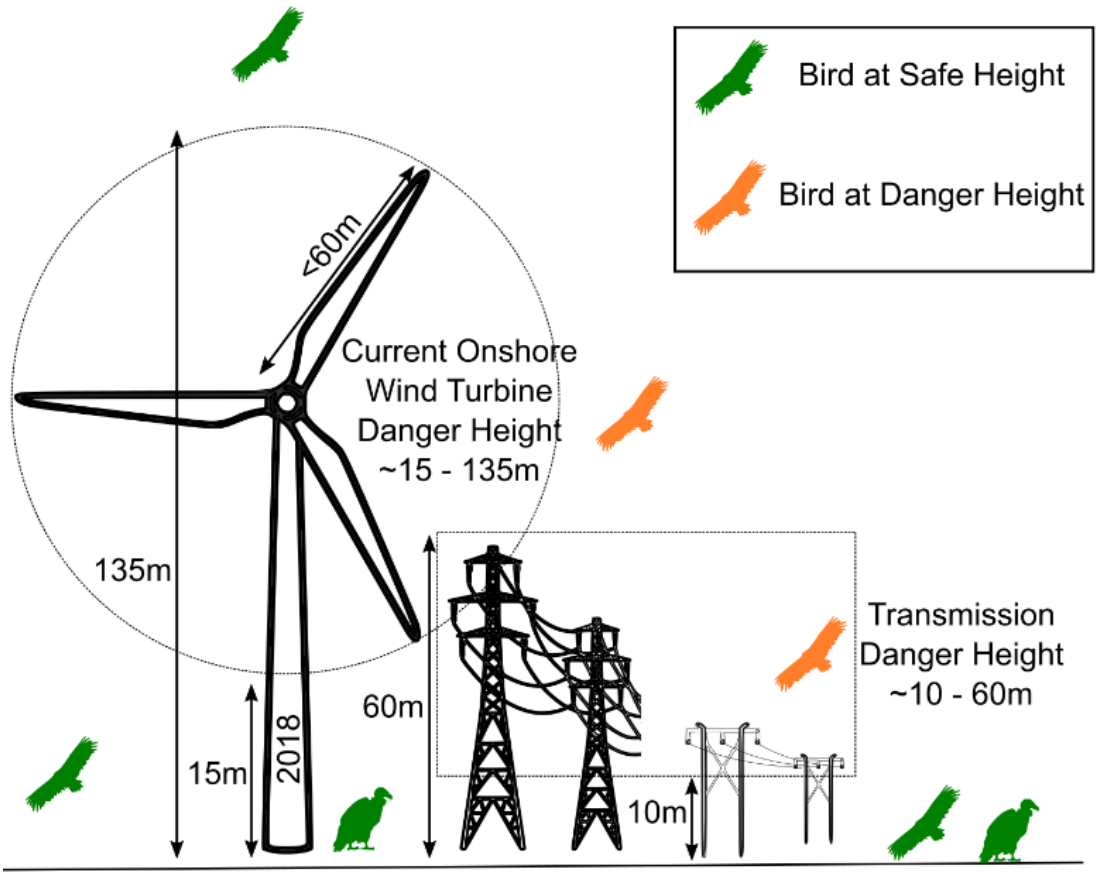


Figure 6.4: Illustration of the danger height bands for different infrastructures. Any bird flying within this zone is potentially at risk of collision.

GPS locations relative to the ground which are more likely to occur when the bird is perched or flying near the ground [PCD+20]. To minimise our reliance on a single metric to define flight we use speed (>1.39m/s) to classify each fix as in flight (1) or not (0) and then exclude outlier GPS locations which fall outwith the 95% confidence interval for height relative to ground. This does not eliminate all GPS locations with a negative flight height however many of these GPS locations will be associated with low altitude flights near ground level and are therefore of relevance to understanding where and when birds are most likely to be at danger height. We then classified all GPS locations in flight at heights between 4m and 135m as within the danger height for collision with contemporary wind turbines [KHKC+18] and power lines [MSH+20, DMÁM+19]. This is shown in Figure 6.4. These were appended to the CSV file before importing to GAMT so that these variables could be seen during visualisation.

Flight height behaviour in relation to landscape factors

To understand how biophysical factors in the landscape influence the likelihood of a bird flying at danger height for each species, we use a generalised linear model (GLM) approach [FRW10] in R. We treated presence at danger height (1) or not (0) as a binomial distribution. Initially we refined the model with a Pearson-Rank coefficient correlation table (Hmisc R package, R Version 4.1.0, [Har18]) to determine most influential factors as measured by Akaike information criterion (AIC) and eliminate co-linear variables as measured by a Variance Inflating Factor analysis ("VIF") index [TRK+18]. In the model we included the following variables as predictors: NDVI [MS19], terrain roughness index (TRI) at 90m resolution derived from STRM-GL1 30m DSM [(SR13], the Land U Component of Wind [MS19] as well as distance (km) from transmission power lines, coastlines, wind turbines, rivers and major roads calculated using a distance raster produced in QGIS using the raster processing toolkit.

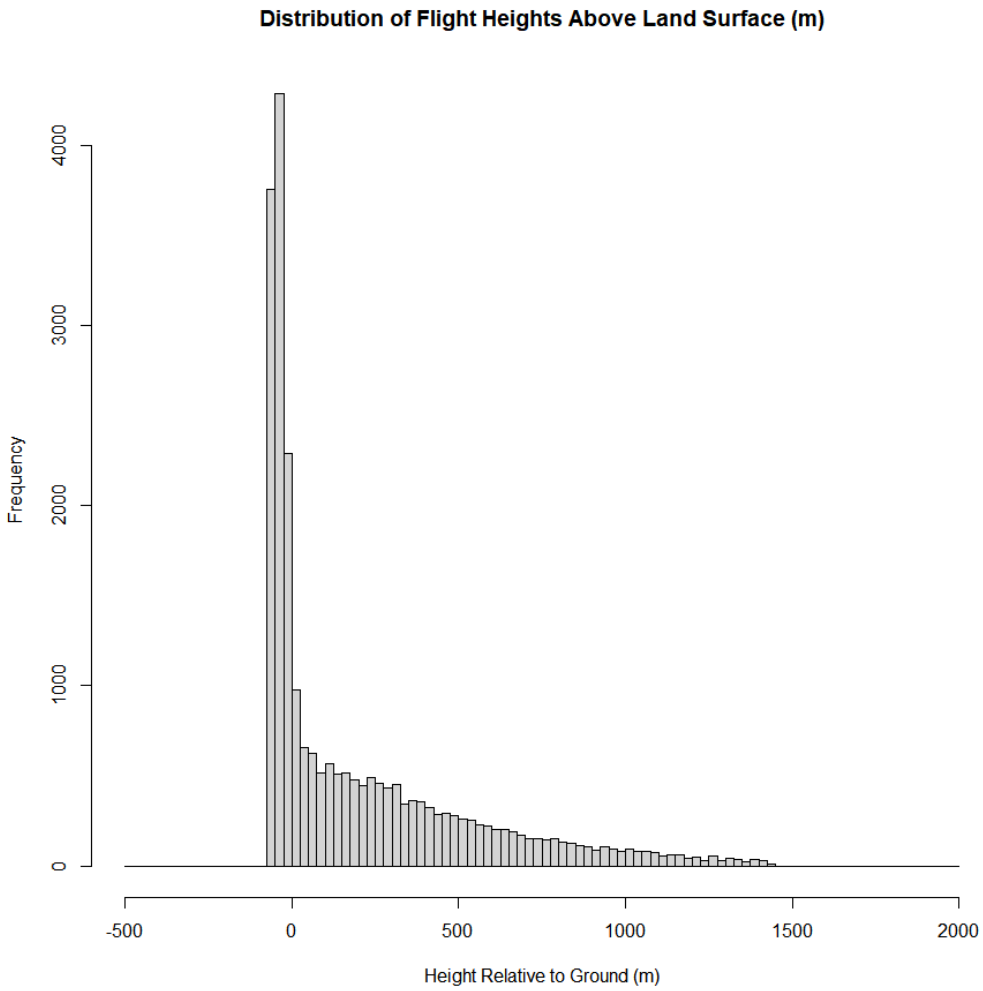


Figure 6.5: Distribution of flight heights across the seven individual storks after removal of outlier heights and locations not in flight associated with speeds less than 1.39m/s.

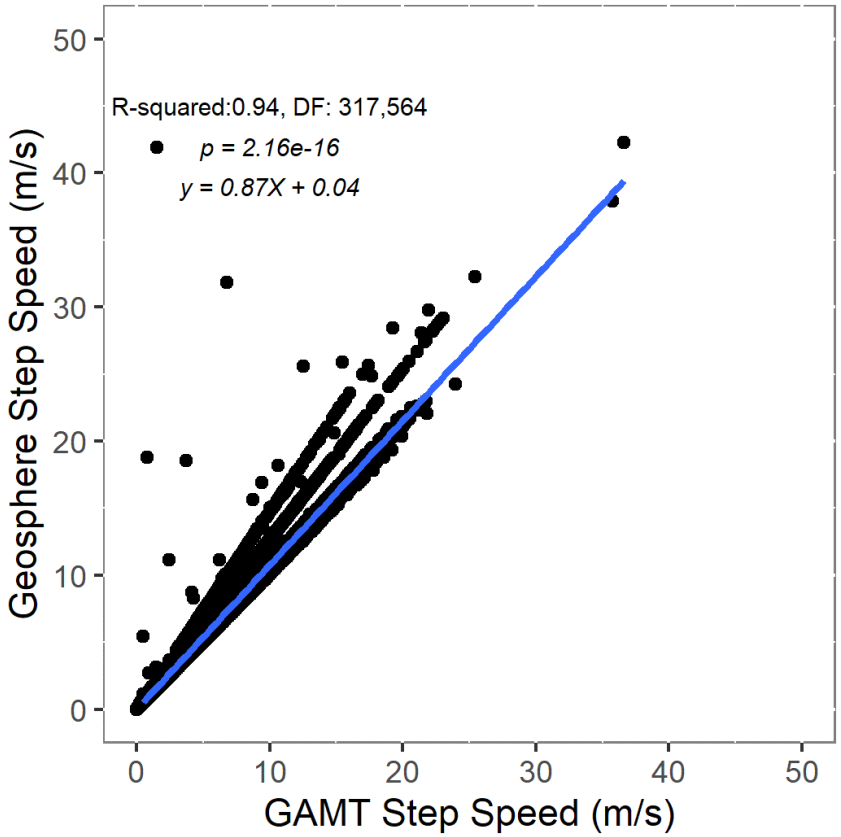


Figure 6.6: Comparison between step speed calculated in GAMT with that calculated in R using the Geosphere package. There is some variation among individuals but there is strong agreement between both methods of calculating step speed. This variation is caused by rounding errors and the use of the faster, but more inaccurate, CUDA trigonometry functions.

Although found to be significant predictors of presence at danger height height, we were unable to include Orographic Uplift or habitat variables at this stage due to issues with Movebank ENV-Tools during our attempts to append the data to the random points further highlighting the potential for GAMT to provide a more reliable method for appending environmental variables onto bird movement data. Going forward for the full flyway scale analysis we will compile our own uplift and habitat raster layers for use with GAMT.

Individual ID	Mean Height (m)	Height 2.5 Percentile (m)	Height 97.5 percentile (m)	Total GPS Locations	Total Locations Associated with Negative Height (m)	Percent at Negative Altitude
O584_Barlavento	185.48	-5	498	5919	445	7.52
O585_Lagos	101.28	-5	434.625	1415	99	6.99
O586_Zambujeiro	71.22	-7	434	3371	605	17.95
O587_Odiaxere	100.44	-6	453.55	2857	340	11.9
O588_Rasmalho	65.31	-7	389.8	2749	456	16.59
O590_Grandaco	121.44	-7	467.8	1528	280	18.32
O591_Bernie	151.98	-9	487	3116	614	19.7

Table 6.2: Summary of individual flight heights relative to ground level for the storks included in this analysis.

Comparison of GAMT and R Packages

Through the use of the R package shown earlier within this chapter, or the GUI within the toolkit itself, it is possible to obtain a number of measures of bird movement regardless of trajectory size. Of key relevance to this work is flight speed as this is important for classifying a record as in flight or not. Here we used a linear regression to compare the flight speed calculated in GAMT with that calculated by using the Geosphere package in R [HWV15].

Point Surface

We use the output from the binomial GLM to predict the proportion of GPS locations at danger height for each grid cell for each month within the study area. To achieve this, we constructed a matrix of random points at a density of 50 points per 5×5 km grid cell ($25km^2$). We appended the variables used in the final model to these points using the Global Animal Movement Toolkit before using the predict function in R ('stats' package version 4.1.0) to calculate the likelihood of them being at danger height on a scale ranging from zero to one. We then calculated the proportion of these points in each grid cell predicted to be at danger height. To produce the sensitivity to collision risk score we combine this with a morpho behavioural conservation status risk index (MBRCI) which accounts for the avoidance ability of the species and the relative impact on the population in terms of their conservation status of a collision mortality

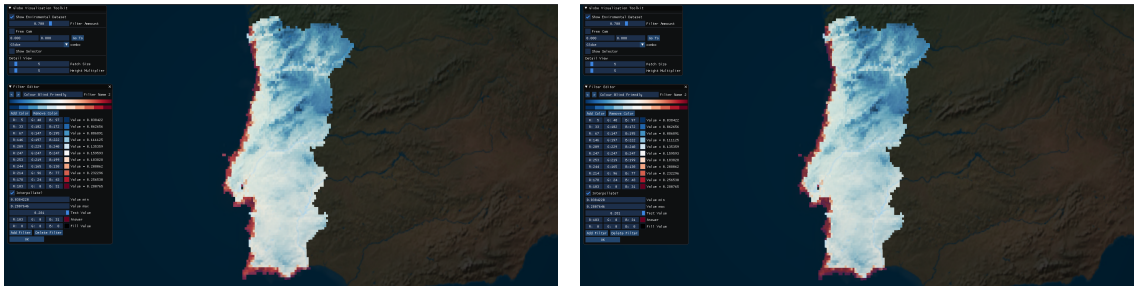
Variable	Estimate	Std. Error	z value	Pr(>|z|)
Intercept	-1.68E+00	3.66E-02	-45.756	<2e-16 ***
Elevation of Ground Surface Relative to Se Level (m)	-1.13E-03	1.68E-04	-6.749	1.49e-11 ***
TRI	2.10E-02	3.14E-03	6.688	2.26e-11 ***
ERA5 Land U Component of Wind	-2.37E-02	1.45E-03	-16.373	<2e-16 ***
Distance to Transmission Lines	-1.51E-03	3.37E-04	-4.462	8.11e-06 ***
Distance to Wind Turbines	1.25E-03	3.86E-04	3.231	0.00123 **

Table 6.3: Results for illustrative Binomial GLM relating environmental factors to the likelihood of storks being present at danger height. Statistical significance is shown as the following: * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

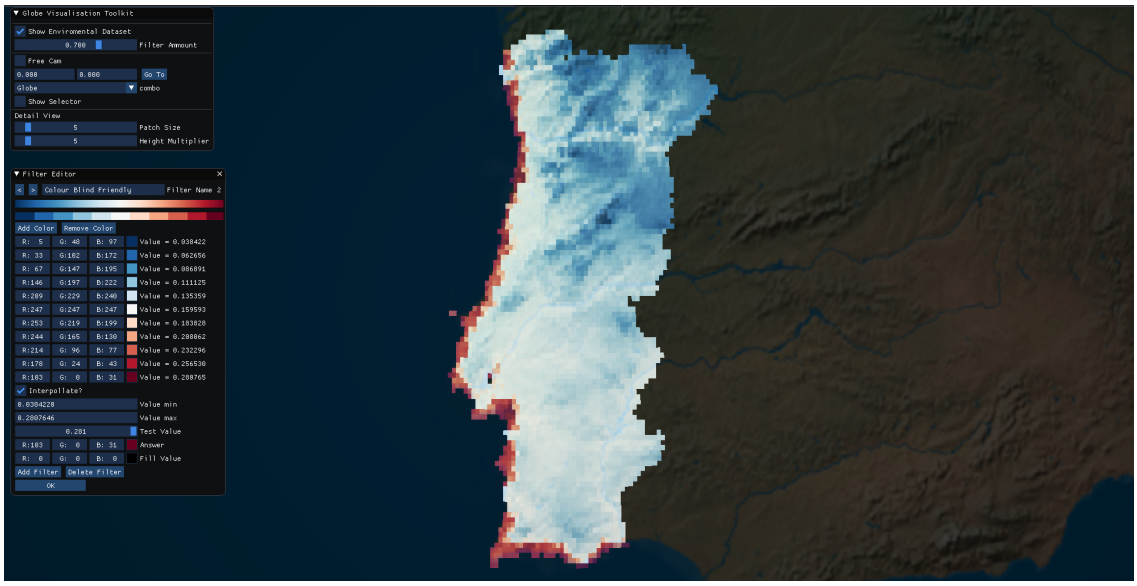
occurring [GSA$^+$ng]. We then visualise this sensitivity surface using GAMT. The results were rasterised using GDAL tools in QGIS [QGI19] and padded out with NoData values to allow for visualisation in GAMT which requires a global extent for raster data inputs.

6.4.2 Results

For the seven adult storks present in the trial dataset used for this case study, linear regression (R2 = 0.94, F(1, 317,564) = 5.335e+06, p = 2.16 * 10-16) reveals that step speed calculation in GAMT compares favourably with using the distHaversine function from the geosphere package to calculate distance between subsequent locations in R and then calculate speed using the time step (Figure 6.6). The reason for the minor differences between the two methods is that the GPU transform uses the lower precision trigonometry functions for performance. These can be changed to the standard trigonometry functions if more accuracy is required at the cost of slightly slower processing time. The key advantage of using GAMT for this is the ease with which primary movement metrics can be calculated for movement data sets, with the ability for users to either do this as part of an R script or in the graphical user interface.

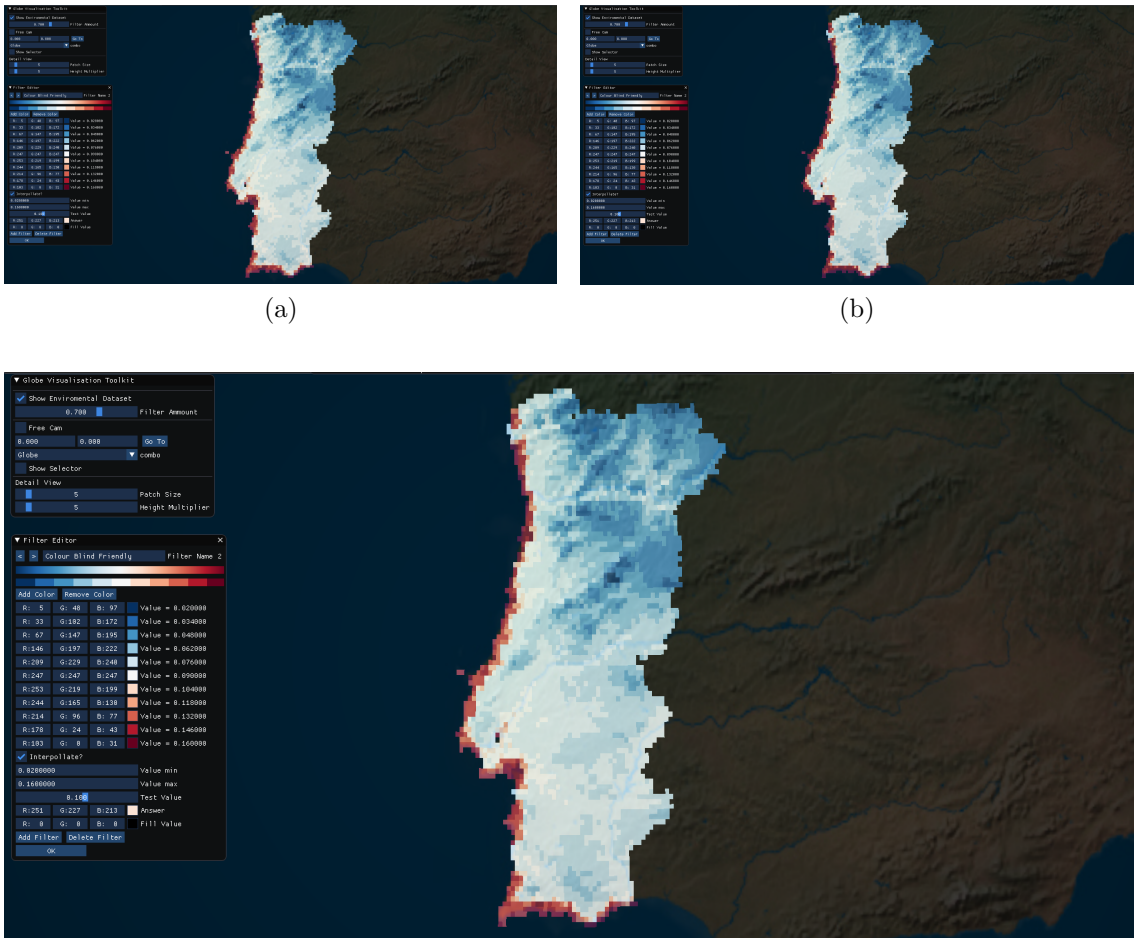


(a) (b)



(c)

Figure 6.7: Predicted proportion of flights at danger height for each 5 × 5km grid cell in Portugal over three different months: (a) Feb, (b) June, (c) Oct). As can be seen, there is a higher predicted proportion of flights at danger height nearer the coasts and at higher elevations (darker colours).



(a) (b)

(c)

Figure 6.8: Predicted sensitivity to collision for each 5 × 5km grid cell in Portugal over three different months: (a) Feb, (b) June, (c) Oct). As can be seen, there is a higher sensitivity to collision risks at the coasts and at higher elevations, denoted by darker colours.

GAMT allows bird movements to be rapidly visualised to help understand bird behaviour at different locations, Figure 6.3a displays the raw tracks and Figure 6.3b displays a space time cube providing a 3D visualisation of the bird movements as they move between breeding and over-wintering areas. For locations classified as in flight, the mean flight height of the 7 individuals ranged between 65.3m and 185.5m (Table 6.2) and the distribution of flight heights is visualised in Figure 6.5.

The correlation table (Appendix A) allowed us to identify the key variables to include in the model while avoiding the use of auto-correlated variables. We included these variables in a binomial GLM (Table 6.4). Unfortunately it was not possible to utilise all of the significant variables because of issues relating to appending variables form Movebank ENV Tools to the random points surface due to server unavailability. We were only able to apply the variables in Table 6.3 to the produce the prediction rasters.

The prediction maps (Figures 6.7 and 6.8) display the predicted proportion of flights at danger height and the predicted sensitivity to collision risks for the months of February, June and October. Based on the variables included in the model these maps highlight how sensitivity to collision is generally higher near the coast and in higher altitude areas. The distribution of sensitivity to collision risk from wind turbines also appears to be similar across different months (Figures 6.9 – 6.11, Table 6.5) however, at this stage these prediction surfaces do not include the likelihood of the birds being present for a given month so do not represent true sensitivity to collision.

6.4.3 Discussion

Here we have shown how GAMT can be used to aid data analysis of large movement data sets by way of the ability to rapidly append relevant environmental variables

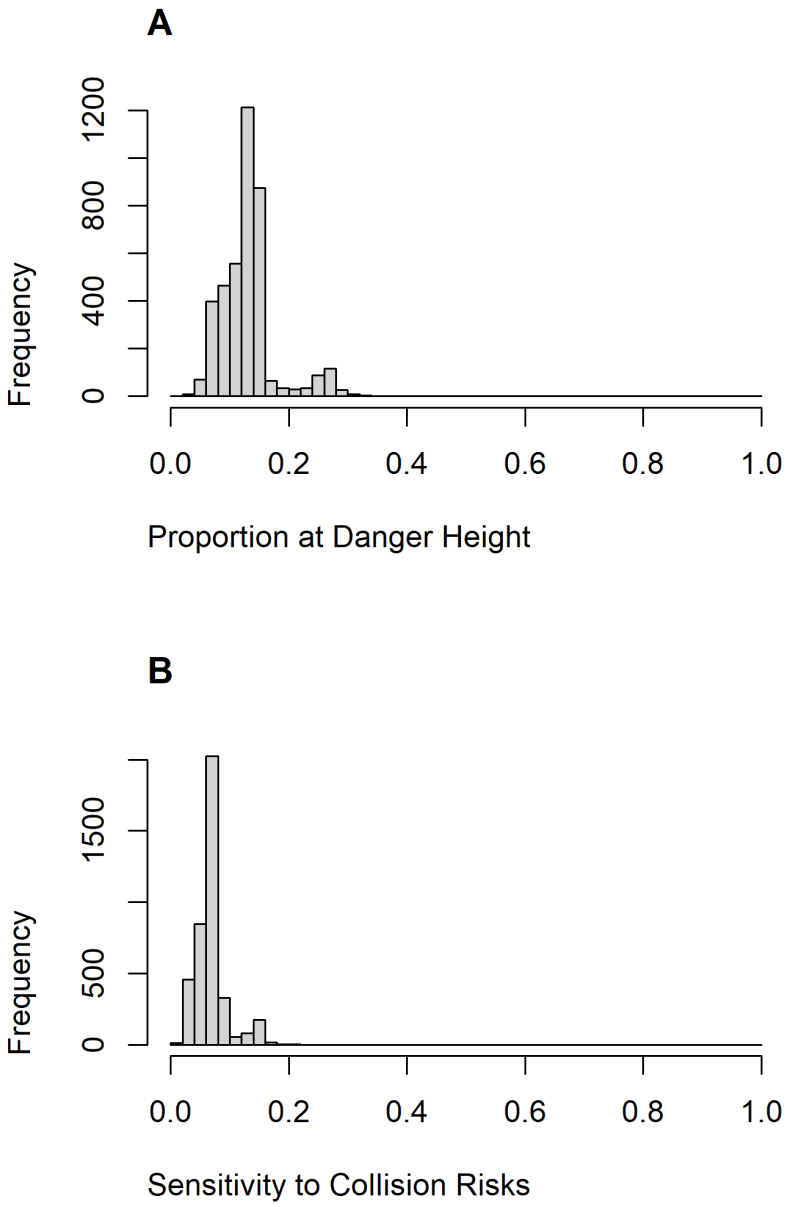


Figure 6.9: A: Distribution of the proportion of flights at danger height for each grid cell in Portugal for the month of February. B: A Distribution of Pseudo-Sensitivity scores across across all grid cells for the month of February calculated by combining the proportion of in flight locations at danger height with the MBRCI score for White Stork Ciconia ciconia.

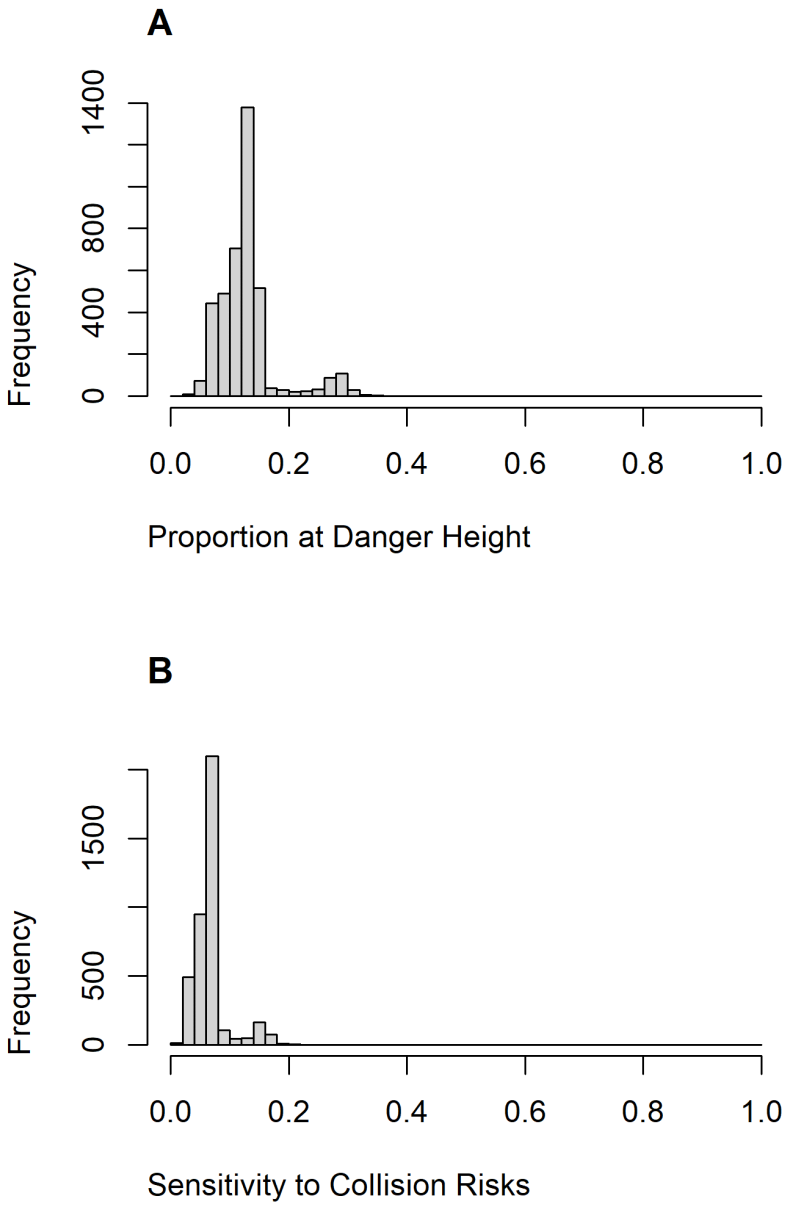


Figure 6.10: A: Distribution of the proportion of flights at danger height for each grid cell in Portugal for the month of June. B: A Distribution of Pseudo-Sensitivity scores across across all grid cells for the month of June calculated by combining the proportion of in flight locations at danger height with the MBRCI score for White Stork Ciconia ciconia.

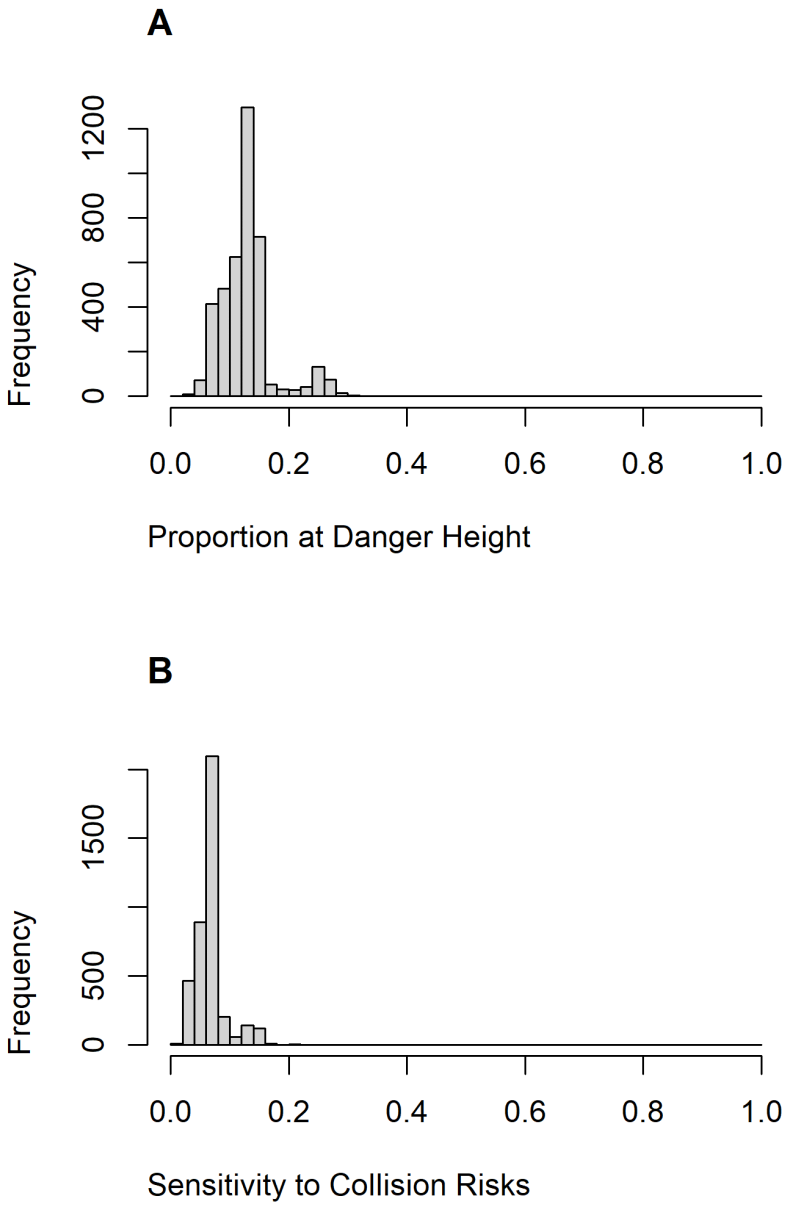


Figure 6.11: A: Distribution of the proportion of flights at danger height for each grid cell in Portugal for the month of October. B: A Distribution of Pseudo-Sensitivity scores across across all grid cells for the month of October calculated by combining the proportion of in flight locations at danger height with the MBRCI score for White Stork Ciconia ciconia.

Variable	Estimate	Std. Error	z value	Pr(>|z|)
Landcover_LCCS1Dense herbaceous	-2.26E+00	1.15E-01	-19.62	<2e-16 ***
Landcover_LCCS1Evergreen broadleaf forests	-1.32E+01	1.97E+02	-0.067	0.946396
Landcover_LCCS1Evergreen needleleaf forest	-1.28E+01	1.97E+02	-0.065	0.948299
Landcover_LCCS1Fill	-2.23E+00	1.00E-01	-22.264	<2e-16 ***
Landcover_LCCS1Sparse forests	-2.29E+00	4.92E-01	-4.648	3.36e-06 ***
Landcover_LCCS1Sparse herbaceous	-9.03E-01	1.01E+00	-0.898	0.369108
Landcover_LCCS1Sparse shrublands	-1.16E+01	1.97E+02	-0.059	0.953085
Land Surface Elevation above Sea Level (m)	-9.69E-04	2.07E-04	-4.679	2.88e-06 ***
Movebank Orographic Uplift	5.10E-01	1.48E-01	3.435	0.000592 ***
TRI	2.42E-02	3.81E-03	6.337	2.35e-10 ***
Percent Tree Cover	2.53E-02	4.23E-03	5.99	2.10e-09 ***
ERA5 Land U Component of Wind	-1.78E-02	1.82E-03	-9.744	<2e-16 ***
NDVI_0_05	7.01E-09	1.82E-09	3.854	0.000116 ***
Distance from Rivers	1.92E-02	4.48E-03	4.275	1.91e-05 ***
Distance from Transmission Lines	-1.84E-03	4.39E-04	-4.196	2.71e-05 ***
Distance from Wind Turbines	3.45E-03	5.21E-04	6.618	3.64e-11 ***
Distance from Coasts	-2.21E-03	6.42E-04	-3.435	0.000592 ***

Table 6.4: Results of the initial Binomial GLM. Statistical significance is shown as the following: * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

and visualise the results. This approach is particularly powerful when combined with statistical packages in R and helps users avoid some of the issues encountered when appending environmental variables using other tools such as Movebank ENV-Tools or sequential CPU processing. The results of this case study also highlight how the movement metrics calculated using GPGPU accelerated methods (step speed in this case) compare well to the results form calculating this via other means with the advantage of being less limited by data set size.

A major limitation of the analysis in this case study was our inability to include all the relevant significant predictors of presence of the birds in flight at danger height such as uplift. As such our results should not be taken as final but more as an example of how GAMT can be used to perform this kind of analysis and then visualise the results. Going forward we plan to produce our own uplift and other raster surfaces to allow them to be appended to data quickly using GAMT. We will also combine the predicted sensitivity to collision with our understanding of the population density

Month	Mean Percent Flights at Danger Height	SD	Min Percent at Danger Height	Max Percent at Danger Height	Mean Sensitivity	SD	Min Sensitivity	Max Sensitivity
February	0.13	0.05	0.03	0.37	0.07	0.03	0.02	0.21
June	0.13	0.05	0.03	0.4	0.07	0.03	0.02	0.22
October	0.13	0.04	0.03	0.36	0.07	0.03	0.02	0.21

Table 6.5: Summary table for the predicted proportion of locations at danger height and sensitivity to collision for each grid cell. Aside from some variation around the maximum, the results are broadly similar between months for the predictions for each grid cell.

for the species used in the analysis to create a spatial data set which can be used by planners and developers in the early stages of site selection for new wind farms to prioritise lower risk areas for birds.

Chapter 7

Conclusions

7.1 Discussion and Conclusions

As the amount of data we collect continues to grow at ever faster rates, it becomes increasingly important that the methods that we apply evolve with this new influx of data. The scalability of the methods we use has become vital. While previous methods that have been optimised for sequential processing have been sufficient in the past, parallel processing has begun to surface to allow for the scalability needed for the new movement datasets now being collected. Other areas of research, such as maritime trajectory visualisation, have begun to capitalise on this [GWD20, HLZL20]. This has led to networked clusters becoming popular to process larger datasets [GWD20]. Within this thesis we have shown that this is not the only avenue for movement ecology to expand. Via the use of GPGPU computing, we have shown that the specialised hardware of distributed computing is not a requirement as we transition to this new era of data collection. While some datasets that contain billions of datapoints may be infeasible to fit onto a single GPU, there is merit expanding the amount of data that a single workstation can process. This enables researchers to visualise and analyse their data in real time, with interactivity playing a large part in the interpretation of these results. We have shown that not only can the visualisation of these larger

movement datasets benefit from GPGPU acceleration, but that with minor changes, existing analytical methods can benefit significantly from this acceleration.

We began by optimising the visualisation methods used by movement ecologists to visualise both trajectory and spatial data. Previous methods relied on CPU sequential rendering methods [SW19] or overlaying of trajectories on a google map interface [XD15]. While suitable for small scale studies, these methods struggled to render large amounts of movement data. MoveVis in particular requires long computation times to complete a single pre-rendered visualisation that needs to be completely recalculated if a change is required [SRSW20]. While standard OpenGL methods may be able to render the large environmental and movement datasets, through the use of GPGPU computing, we have shown that it is possible to interactively render environmental datasets containing millions of datapoints in real time. CUDA streams also enable us to dynamically switch between multiple global environmental datasets, allowing for a multitude of datasets to be considered together. This can also be used to switch between timestamps of the same dataset, allowing for changes over time to be considered. These optimisations also allow for lower end GPUs to be used, with them performing similar to high end ones. There are limits to this approach, however. While not an issue for higher end GPUs, the smaller memory available to GPUs when compared to RAM can become an issue as the resolution of global environmental datasets becomes too high. Also, as of writing there is no tiling support for environmental datasets. While this can be solved by first combining the tiles through the use of an outside package, it is an inconvenient extra step when visualising large amounts of data.

With interactive frame rates achieved for visualisation, in Chapter 4 we move to accelerating the analysis of the trajectory data. With repositories of trajectory data growing ever larger [Mro18] we apply GPU accelerated parallel methods to enable

interactive computation. This interactive computation vastly reduces the disconnect between parameter adjustment and results, with the effect of parameter change being immediately available. While it may be more intuitive to design analytical methods that utilise sequential computing, we show that with only minor adjustments the same results can be achieved with more efficient scaling to global datasets. With a number of GPU programming APIs now available [WLB+20, Gui13], it is now possible for the application of parallel primitives to be used in the design of parallel methods without the necessary computer graphics or GPU programming background. We show this by first implementing a number of primary and secondary order movement metrics [SDCG18] that describe the basic parameters of movement. These movement metrics can then be reduced to calculate summary statistics of movement. With these movement metrics calculated in a fraction of a second, they can be used in more involved techniques to analyse the movement of a species.

We first apply this by utilising stream compaction to threshold metrics to identify when a movement metric passes a user defined threshold. This allows for the detection of events within a trajectory that may indicate a decision within movement. We also show that partial reduction can be used to detect the start and end of days, that can then be used to calculate a number of daily statistics, such as daily movement, daily net movement and placing an event at the start and end of each day. We further this by then implementing a GPGPU accelerated version of the turning point detection algorithm by Potts et al [PBS+18]. While this method has been optimised for sequential processing on the CPU, GPU acceleration allows for even more efficient processing of large amounts of data. With this method specifically designed for high resolution data, the GPU accelerated version allows for higher scalability when these high resolution datasets are recorded for longer periods of time. We also implemented two migration detection algorithms highlighted by Soriano-Redondo et al [SAF+20],

spatial thresholding and absolute displacement. Through a combination of parallel primitives it is possible to detect the beginning and end of a migration in real time. It should be noted that these methods are highly sensitive to parameter adjustment, with the ideal parameters likely unknown before processing. The real time calculation of these methods allows for researchers to perform these algorithms multiple times in quick succession to experiment and investigate the parameter adjustments made.

Some of these methods require a regular sampling interval however [PBS+18]. It is unrealistic to assume that there are no gaps during data collection. For example, canopy cover, hardware malfunction and lack of signal have been shown to cause significant gaps in data collection [Lon16]. Resampling of trajectory data is therefore an important step when pre-processing data. There are a large number of different interpolation techniques used when resampling data [Lon16]. We implement a GPGPU accelerated parallel resampling algorithm for trajectory data that can be used to apply a resampling method of choice. We implement both linear interpolation and Catmul-Rom splines as examples. By calculating these in real time they can be used as a pre-processing step for other algorithms. It is also possible to resample the same dataset to differing sampling intervals and investigate the effect. It should be noted that the amount of memory available on a GPU does limit the amount of data that can be processed and output, with system memory typically able to hold more. With newer cards, however, if a single trajectory is processed at a time, there should be more than enough memory. For example, each relocation of a trajectory requires 40 bytes in memory, 24 bytes for the position (double precision longitude, latitude and altitude) and 16 bytes for the timestamp. This means that if 1 Gigabyte of GPU memory is free (not in use by the visualisation) it can store approximately 26.6 million relocations. Assuming a sampling interval of 1 minute, this results in 51 years of trajectory data that can be stored. This is plenty for a single trajectory, however if

multiple trajectories that contain multiple years of data are processed simultaneously, memory could become an issue.

With large amounts of data being processed and visualised together, it becomes impractical to attempt to visualise the large amounts of trajectory data simultaneously. Occlusion of data and other issues begin to arise if we attempt to do so. To rectify this, in Chapter 5 we have implemented a flow map generalisation method by Andrienko and Andrienko [AA11]. While similar methods have recently been developed for large scale maritime visualisation [GWD20], these methods often use networked clusters. By modifying the method by Andrienko and Andrienko to better suit GPGPU computing, we have shown that these flow maps can be calculated interactively. With the sequential CPU method, the clustering of event points tends towards $O(n^2)$ as the number of points within each cluster increases. Conversely, we implement a GPGPU accelerated canopy clustering method that tends towards $O(n)$ as the number of points within each cluster increases. This has the effect of higher levels of generalisation requiring more computation with the CPU method, and less so for the GPGPU accelerated methods, which we argue is more intuitive. We then combine the output of these flow maps with other visualisation techniques to filter out unneeded data. For example, space time cubes suffer from confusing visualisations as the number of trajectories increases. By filtering the visible trajectories to only the selected flows large amounts of the visual clutter is cleared, while also clarifying the position of the trajectories to the selected areas. This allows for the control of obfuscation that typically occurs when generalising large amounts of data [GG13]. These space time cubes, again calculated using GPGPU acceleration, can be adjusted in real time. Both the camera position and the parameters of the space time cube (position, start date and end date) can be adjusted to assist in investigation. These flow maps can also be visualised with digital elevation data to show height data, or

on the surface of a globe to minimise distortion of distances over large areas.

With the repositories of R packages continuing to grow, more diverse methods are beginning to be integrated into the work flow of movement ecologists [JBC+19]. It is important that any new methods published can be used in tandem with existing packages, both for ease of learning and for ease of integration into existing studies. To this end, within Chapter 6 we implement the Global Animal Movement Toolkit (GAMT) and GAMTr. GAMT is a C++ desktop application allowing researchers to utilise the GPGPU accelerated methods without the need to understand the intricacies of GPU programming. We show that the GUI within GAMT allows for the import of CSV files containing trajectory data that can then be visualised in real time using all of the previous methods described within this thesis. Researchers can separate their movement data into studies to allow for clearer visualisations and to process data together. With a GUI interface the user can immediately see the results of their analysis, be it metric calculation or event detection. Any annotation that occurs can then be exported for use within other R packages.

To facilitate the integration of GAMT with current and future studies, we have also implemented GAMTr. GAMTr is a R package that acts as an intermediate between the C++ application and R environment. This R package enables for a visual analytics interface for their data as they process it. Any dataframe that contains trajectories can be imported via an R command, instead of the GAMT GUI, which will then be converted to a suitable format. This can then be visualised and explored as if the user had imported a CSV file. Any visualisation and analytical methods can then be performed, with any metrics and environmental variables appended to the dataframe upon export. This exported dataframe can then be used with any other of the R packages desired. GAMTr also supports the automated download of commonly used environmental datasets. This is implemented using the Copernicus

Data Store (CDS) API within R. At any point the user can specify an environmental variable, for example surface temperature, and the desired timestamps and GAMTr will download and immediately visualise the data within the toolkit. Researchers can also create their own environmental dataset raster and import it into the toolkit for visualisation. With both environmental and movement datasets able to be imported with ease, researchers can visualise and explore their data with minimal delay.

Throughout the development of methods within this thesis, the school of environmental sciences at UEA have had extensive input. All of the methods contained within have been in response to ongoing research into the movement ecology of a number of different species [GCS+16b, GSA+ng, GCS+16a, SAF+20]. While a dedicated user study chapter was not possible due to lockdown restrictions due to COVID, feedback and improvement of the visualisation and analysis methods has guided development throughout, for example the case study within Chapter 6.

While maritime data is beginning to utilise the power of GPU acceleration, with many of the techniques also applicable to other movement datasets [HLZL20], it is hoped that the culmination of this research results in more GPU application within the field of movement ecology. With dataset sizes on the rise, it is important that we begin the switch to more scalable methods to ensure efficient processing. While there are now online repositories that allow for networked processing of movement data [Mro18, DBW+13], these can fail, or simply be unavailable. With GPU programming APIs becoming more commonplace [Sil86, MGM+11], we can begin to utilise the power of the hardware locally available to process data in real time. Through the use of GPU parallel primitives, most algorithms can be implemented and processed significantly faster, often with minor adjustments, resulting in processing times orders of magnitude quicker.

7.2 Future Work

With a number of GPGPU accelerated methods now available, there is room for further implementation to allow for the ease of accessibility to these methods and further studies that could benefit from GPGPU acceleration. These are as follows:

7.2.1 Integration of methods with existing GIS.

While there are features lacking in terms of previously implemented methods, such as tiling support, the focus of this research was more for GPGPU accelerated techniques than reimplementing a GIS for movement ecology in its entirety. With the rise of open source GIS, such as QGIS, and the ability for plugins to be shared, the methods here could instead be implemented as a series of plugins, rather than a stand alone toolkit. This would enable researchers to integrate these GPGPU accelerated methods within their workflows with ease. CUDA plugins have begun to be developed for QGIS, such as CUDA Raster [FKH], but these are early in development. The combination of GPGPU accelerated techniques, alongside a highly developed GIS environment would be a powerful tool for movement ecologists. This would also avoid the need to reimplement any underlying methods that are already available within QGIS as standard, while also allowing for support of future datasets with minimal changes to the GPGPU accelerated methods.

7.2.2 GPGPU acceleration of Data cleaning and other Movement ecology techniques.

There are still a plethora of movement ecology techniques that can benefit from GPGPU acceleration. These include, but are not limited to, Habitat selection analysis and its extension integrated step selection analysis [APLB16]. Many data cleaning

methods, such as those included within the amt package within R [SFA19], could lend themselves well to GPGPU acceleration that alongside the resampling method within Chapter 4 would allow for large amounts of data to be cleaned before visualisation and analysis. Many home range estimation techniques could be accelerated. GPGPU accelerated kernel density based methods already exist within literature [ZZH17], and with some minor adjustment to compensate for geodetic distances, could allow for the home range estimation [FC17], migration detection [SAF$^+$20], and visualisation of event points. As methods within movement ecology progress, it is hoped that newer methods being developed make use of parallel processing through the use of parallel primitives, with scalability of these newer methods being considered.

Appendix A

Pearson-Rank coefficient correlation tables

	event_id	Aspect	Distance to Coast (km)	Danger Height (0 or 1)	Distance Between Locations (km)	Elevation Relative to Elispoid (m)	Elevation relative to Mean Sea Level (m)	Hour of Day	Year	Month	Time between GPS locations (Minutes)
Aspect	0.01										
Distance to Coast (km)	-0.07****	-0.03****									
Danger Height (0 or 1)	-0.04****	0.01	-0.08****								
Distance Between Locations (km)	0.23****	0.01	0.09****	0							
Elevation Relative to Elispoid (m)	-0.12****	-0.03****	0.33****	-0.07****	0.03****						
Elevation relative to Mean Sea Level (m)	-0.12****	-0.03****	0.33****	-0.07****	0.03****	1.00****					
Hour of Day	0.04****	0.01	0	0.01	0.10****	0	0				
Year	0.92****	0.02**	-0.12****	-0.01	0.19****	-0.04****	-0.04****	0.05****			
Month	0.07****	-0.03****	0.11****	-0.07****	0.06****	-0.18****	-0.18****	-0.03****	-0.32****		
Time between GPS locations (Minutes)	0.39****	0.01*	-0.07****	0	0.57****	-0.09****	-0.09****	0.03****	0.36****	0.07****	
Step Speed (m/s)	-0.07****	0	0.20****	-0.01	0.60****	0.14****	0.14****	0.12****	-0.08****	0.01	-0.12****
Heading	-0.02*	-0.01	-0.04****	-0.01*	0	0	0	0.02**	-0.03****	0.04****	0
NID (Grid Square ID)	0.11****	-0.04****	0.66****	-0.12****	-0.09****	0.44****	0.44****	-0.02**	0.06****	0.14****	-0.10****
location_long	-0.21****	0	0.32****	0.04****	-0.02***	0.29****	0.29****	-0.10****	-0.13****	-0.15****	-0.03****
location_lat	-0.13****	0.02***	-0.47****	0.09****	-0.09****	0.02**	0.02**	-0.08****	0	-0.29****	0.04****
Movebank_Thermal Uplift_ECMWF	0.01	0	0.30****	-0.02**	0.09****	0.29****	0.29****	-0.13****	-0.04****	0.08****	-0.11****
tag_id	-0.19****	-0.01	0.11****	0.01	-0.03****	0.13****	0.13****	-0.08****	-0.15****	-0.07****	0.02**
TRI	0	0.02*	-0.10****	0.12****	0.05****	-0.07****	-0.07****	0.01	0	0	0.06****
Distance from a River	-0.05****	-0.03****	0.08****	0.03****	0.07****	0.21****	0.21****	0.01	-0.01	-0.12****	-0.04****
Distance from a Road	0	-0.01	0.42****	-0.02***	0.09****	0.14****	0.14****	0.04****	0.01*	-0.06****	-0.03****
Distance from Transmission Power Line	0.06****	-0.02**	0.63****	-0.05****	0.12****	0.07****	0.07****	0.08****	0.09****	-0.12****	-0.02***
Distance from Wind Turbine	0.15****	-0.02***	0.63****	-0.10****	0.10****	-0.01	-0.01	0.10****	0.03****	0.26****	-0.04****
Monthly_1km_NDVI	-0.03****	0.03****	-0.43****	0.10****	-0.07****	-0.29****	-0.29****	-0.07****	0.05****	-0.16****	0.08****
NDVI_0_05	-0.03****	0.03****	-0.44****	0.08****	-0.06****	-0.29****	-0.29****	-0.06****	0.06****	-0.18****	0.08****
ERA5.Land.2m.Temperature	0	-0.03****	0.26****	-0.19****	-0.07****	0.18****	0.18****	0.06****	-0.06****	0.14****	-0.09****
ERA5.Land.Surface.Pressure	0	-0.03****	0.16****	-0.18****	-0.11****	0.11****	0.11****	-0.01*	-0.02***	0.06****	-0.08****
ERA5.Land.U.Component.of.Wind	-0.03****	-0.02***	0.16****	-0.17****	-0.11****	0.22****	0.22****	0.02*	-0.05****	0.05****	-0.08****
ERA5.Land.V.Component.of.Wind	-0.02**	-0.03****	0.20****	-0.18****	-0.10****	0.21****	0.21****	-0.02**	-0.04****	0.05****	-0.09****
ERA5.Total.Cloud.Cover	0.04****	0	-0.03****	-0.02*	-0.01	-0.07****	-0.07****	-0.07****	0.04****	-0.01	0.02**
ERA5.Total.Precipitation	-0.03****	0.02***	-0.03****	-0.01	-0.02**	0.04****	0.04****	0.05****	-0.02**	-0.02*	0.01
MODIS.Land.Cover.500m. Yearly.Combined. FAO.LCCS1.Land.Cover	-0.43****	0.01	-0.11****	0.02**	-0.10****	0.01	0.01	-0.05****	-0.38****	-0.10****	-0.12****
MODIS.Land.Cover.500m. Yearly.Combined. FAO.LCCS2.Land.Use	-0.42****	0.01	-0.10****	0.02**	-0.10****	0.02***	0.02***	-0.05****	-0.38****	-0.09****	-0.11****
MODIS.Land.VCF.250m. Yearly.Terra. Percent.Tree.Cover	-0.39****	0.01	-0.17****	0.08****	-0.09****	-0.06****	-0.06****	0	-0.34****	-0.07****	-0.11****
Movebank.Orographic.Uplift. from.SRTM.DEM.and.ECMWF.	0.01	0.16****	-0.10****	0.06****	-0.01	0.01	0.01	0.08****	0.04****	-0.06****	0.03***
Height Above Ground (m)	0.06****	-0.03****	0.31****	-0.18****	0.24****	0.20****	0.20****	0.20****	0.01	0.10****	-0.06****

Table A.1: The Pearson-Rank coefficient correlation table used to determine most influential factors as measured by Akaike information criterion and eliminate co-linear variables as measured by a Variance Inflating Factor analysis ("VIF") index. Part 1.

	Step Speed (m/s)	Heading	NID (Grid Square ID)	location_long	location_lat	Movebank_Thermal Uplift_ECMWF	tag_id	TRI	Distance from a River	Distance from a Road	Distance from Transmission Power Line	Distance from Wind Turbine
Heading	-0.02***											
NID (Grid Square ID)	-0.04****	0.04****										
location_long	0	0.04****	0.55****									
location_lat	-0.17****	0.07****	0.99****	0.61****								
Movebank_Thermal Uplift_ECMWF	0.16****	0	0.39****	0.13****	-0.17****							
tag_id	-0.05****	0.03****	0.20****	0.58****	0.44****	0.18****						
TRI	0	0.02**	-0.21****	0.07****	0.14****	-0.12****	0.02***					
Distance from a River	0.14****	-0.02***	0.01	0.01	-0.05****	0.11****	0.12****	0				
Distance from a Road	0.17****	-0.05****	0.35****	0.04****	-0.16****	0.13****	-0.04****	-0.04****	0.04****			
Distance from Transmission Power Line	0.18****	-0.05****	0.58****	-0.64****	-0.88****	0.14****	-0.24****	-0.05****	0.06****	0.52****		
Distance from Wind Turbine	0.19****	-0.07****	0.61****	-0.80****	-0.97****	0.15****	-0.49****	-0.15****	0.03****	0.23****	0.87****	
Monthly_1km_NDVI	-0.17****	0.05****	0.01	0.09****	0.39****	-0.25****	0.10****	0.06****	-0.04****	-0.31****	-0.48****	-0.40****
NDVI_0_05	-0.17****	0.05****	-0.06****	0.13****	0.44****	-0.29****	0.10****	0.06****	-0.11****	-0.31****	-0.47****	-0.44****
ERA5.Land.2m.Temperature	-0.01	0	0.41****	-0.15****	-0.34****	0.27****	-0.03****	-0.35****	-0.19****	0.12****	0.17****	0.35****
ERA5.Land.Surface.Pressure	-0.07****	0.01	0.41****	-0.13****	-0.24****	0.20****	-0.01*	-0.36****	-0.22****	0.08****	0.12****	0.26****
ERA5.Land.U.Component.of.Wind	-0.07****	0.02***	0.41****	-0.03****	-0.13****	0.21****	0.04****	-0.34****	-0.21****	0.07****	0.06****	0.16****
ERA5.Land.V.Component.of.Wind	-0.04****	0	0.42****	-0.07****	-0.23****	0.22****	0	-0.35****	-0.17****	0.09****	0.11****	0.24****
ERA5.Total.Cloud.Cover	-0.03****	0	0.04****	-0.14****	-0.10****	-0.01	-0.07****	-0.04****	0.03****	-0.01	0.02*	0.10****
ERA5.Total.Precipitation	-0.02***	0	0.05****	0.04****	0.06****	-0.04****	0.04****	-0.03****	-0.01*	-0.03****	-0.04****	-0.05****
MODIS.Land.Cover.500m. Yearly.Combined. FAO.LCCS1.Land.Cover	-0.04****	0.03****	-0.03****	0.10****	0.22****	-0.01	0.15****	0	0.14****	-0.07****	-0.10****	-0.22****
MODIS.Land.Cover.500m. Yearly.Combined. FAO.LCCS2.Land.Use	-0.03****	0.03****	-0.01	0.10****	0.21****	0.01	0.16****	0	0.16****	-0.07****	-0.10****	-0.21****
MODIS.Land.VCF.250m. Yearly.Terra. Percent.Tree.Cover	-0.03****	0.03****	-0.10****	0.16****	0.28****	-0.14****	0.10****	0.12****	-0.01	-0.09****	-0.15****	-0.28****
Movebank.Orographic.Uplift. from.SRTM.DEM.and.ECMWF.	-0.02**	0.01	-0.12****	0.05****	0.15****	-0.06****	-0.03****	0.05****	-0.07****	-0.05****	-0.08****	-0.15****
Height Above Ground (m)	0.38****	-0.05****	0.09****	-0.19****	-0.44****	0.28****	-0.17****	-0.08****	0.11****	0.20****	0.34****	0.46****

Table A.2: The Pearson-Rank coefficient correlation table used to determine most influential factors as measured by Akaike information criterion and eliminate co-linear variables as measured by a Variance Inflating Factor analysis ("VIF") index. Part 2.

	Monthly_1km_NDVI	NDVI_0_05	ERA5.Land. 2m.Temperature	ERA5.Land. Surface.Pressure	ERA5.Land U.Component.of.Wind	ERA5.Land V.Component.of.Wind	ERA5. Total.Cloud.Cover	ERA5. Total.Precipitation	MODIS. Land.Cover.500m. Yearly.Combined FAO.LCCS1 Land.Cover	MODIS Land.Cover.500m. Yearly.Combined FAO.LCCS2 Land.Use	MODIS Land.VCF.250m Yearly.Terra.Percent .Tree.Cover	Movebank Orographic.Uplift from.SRTM.DEM and.ECMWF.	Movebank Thermal.Uplift from.ECMWF.
NDVI_0_05	0.90****												
ERA5.Land.2m.Temperature	-0.25****	-0.23****											
ERA5.Land.Surface.Pressure	-0.16****	-0.13****	0.97****										
ERA5.Land. U.Component.of.Wind	-0.14****	-0.10****	0.94****	0.96****									
ERA5.Land V.Component.of.Wind	-0.16****	-0.14****	0.95****	0.96****	0.94****								
ERA5.Total.Cloud.Cover	0.12****	0.12****	0.02**	0.05****	0.05****	0.07****							
ERA5.Total.Precipitation	0.07****	0.08****	0.04****	0.07****	0.10****	0.09****	0.13****						
MODIS.Land.Cover.500m. Yearly.Combined. FAO.LCCS1.Land.Cover	0.10****	0.05****	-0.07****	-0.04****	-0.02*	-0.03****	0.11****	0.07****					
MODIS.Land.Cover.500m. Yearly.Combined. FAO.LCCS2.Land.Use	0.09****	0.04****	-0.07****	-0.05****	-0.02**	-0.03****	0.11****	0.08****	0.99****				
MODIS.Land.VCF.250m. Yearly.Terra. Percent.Tree.Cover	0.28****	0.22****	-0.16****	-0.12****	-0.10****	-0.12****	-0.04****	0.01	0.23****	0.22****			
Movebank. Orographic.Uplift. from.SRTM.DEM and.ECMWF.	0.08****	0.10****	-0.05****	-0.04****	0	-0.05****	0.02*	0.03****	0.03****	0.03**	0.14****		
Height Above Ground (m)	-0.34****	-0.36****	0.17****	0.06****	0.03****	0.08****	-0.06****	-0.05****	-0.16****	-0.15****	-0.12****	-0.06****	0.14****

Table A.3: The Pearson-Rank coefficient correlation table used to determine most influential factors as measured by Akaike information criterion and eliminate co-linear variables as measured by a Variance Inflating Factor analysis ("VIF") index. Part 3.

Bibliography

[AA07] Natalia Andrienko and Gennady Andrienko. Designing visual analytics methods for massive collections of movement data. *Cartographica*, 42(2):117–138, jun 2007.

[AA11] Natalia Adrienko and Gennady Andrienko. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on Visualization and Computer Graphics*, 17(2):205–219, 2011.

[AA13] Natalia Andrienko and Gennady Andrienko. Visual analytics of movement: An overview of methods, tools and procedures, jan 2013.

[AAC+17a] Natalia Andrienko, Gennady Andrienko, Elena Camossi, Christophe Claramunt, Jose Manuel Cordero Garcia, Georg Fuchs, Melita Hadzagic, Anne Laure Jousselme, Cyril Ray, David Scarlatti, and George Vouros. Visual exploration of movement and event data with interactive time masks. *Visual Informatics*, 1(1):25–39, mar 2017.

[AAC+17b] Natalia Andrienko, Gennady Andrienko, Elena Camossi, Christophe Claramunt, Jose Manuel Cordero Garcia, Georg Fuchs, Melita Hadzagic, Anne-Laure Jousselme, Cyril Ray, David Scarlatti, et al. Visual exploration of movement and event data with interactive time masks. *Visual Informatics*, 1(1):25–39, 2017.

[AAD+10] Gennady Andrienko, Natalia Andrienko, Urska Demsar, Doris Dransch, Jason Dykes, Sara Irina Fabrikant, Mikael Jern, Menno-Jan Kraak, Heidrun Schumann, and Christian Tominski. Space, time and visual analytics. *International Journal of Geographical Information Science*, 24(10):1577–1600, oct 2010.

[AAH+13] Gennady Andrienko, Natalia Andrienko, Christophe Hurter, Salvatore Rinzivillo, and Stefan Wrobel. Scalable analysis of movement data for extracting and exploring significant places. *IEEE Transactions on Visualization and Computer Graphics*, 19(7):1078–1094, 2013.

[AAV03] Gennady Andrienko, Natalia Andrienko, and Hans Voss. Gis for everyone: the commongis project and beyond. In *Maps and the Internet*, pages 131–146. Elsevier, 2003.

[ACHLRZ14] Jesús Alvarez-Cedillo, Juan Herrera-Lozada, and Israel Rivera-Zarate. Implementation strategy of NDVI algorithm with nvidia thrust. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8333 LNCS, pages 184–193. Springer Verlag, oct 2014.

[Ama09] C. Amante. ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis, 2009. type: dataset.

[ANZ03] Daniel Adler, Oleg Nenadic, and Walter Zucchini. Rgl: A r-library for 3d visualization with opengl. In *Proceedings of the 35th Symposium of the Interface: Computing Science and Statistics, Salt Lake City*, volume 35, pages 1–11, 2003.

[APLB16] Tal Avgar, Jonathan R. Potts, Mark A. Lewis, and Mark S. Boyce. Integrated step selection analysis: Bridging the gap between resource selection and animal movement. *Methods in Ecology and Evolution*, 7(5):619–630, may 2016.

[ARH+15] Fereshteh Amini, Sebastien Rufiange, Zahid Hossain, Quentin Ventura, Pourang Irani, and Michael J. McGuffin. The impact of interactivity on comprehending 2D and 3D visualizations of movement data. *IEEE Transactions on Visualization and Computer Graphics*, 21(1):122–135, jan 2015.

[BAB+17] Karl Bladin, Emil Axelsson, Erik Broberg, Carter Emmart, Patric Ljung, Alexander Bock, and Anders Ynnerman. Globe browsing: Contextualized spatio-temporal planetary surface visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):802–811, 2017.

[Bat81] Edward Batschelet. Circular statistics in biology. *ACADEMIC PRESS, 111 FIFTH AVE., NEW YORK, NY 10003, 1981, 388*, 1981.

[BBB95] Richard H Bartels, John C Beatty, and Brian A Barsky. *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, 1995.

[BBB+18] J Bernardino, K Bevanger, R Barrientos, J F Dwyer, A T Marques, R C Martins, J M Shaw, J P Silva, and F Moreira. Bird collisions with power lines: State of the art and priority areas for research. *Biological Conservation*, 222(March):1 – 13, 2018.

[BCD17] Robert J. Burnside, Nigel J. Collar, and Paul M. Dolman. Comparative migration strategies of wild and captive-bred asian houbara chlamydotis macqueenii. *Ibis*, 159(2):374–389, 2017.

[BCGL+14] Owen R Bidder, Hamish A Campbell, Agustina Gómez-Laich, Patricia Urgé, James Walker, Yuzhi Cai, Lianli Gao, Flavio Quintana, and Rory P Wilson. Love thy neighbour: automatic animal behavioural classification of acceleration data using the k-nearest neighbour algorithm. *PloS one*, 9(2):e88609, 2014.

[BCL14] Mathias Bourgoin, Emmanuel Chailloux, and Jean-Luc Lamotte. Efficient abstractions for gpgpu programming. *International Journal of Parallel Programming*, 42(4):583–600, 2014.

[BFUY14] Sarah E Battersby, Michael P Finn, E Lynn Usery, and Kristina H Yamamoto. Implications of web mercator and its use in online mapping. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 49(2):85–101, 2014.

[BGKV17] Eric T. Bradlow, Manish Gangwar, Praveen Kopalle, and Sudhir Voleti. The Role of Big Data and Predictive Analytics in Retailing. *Journal of Retailing*, 93(1):79–95, mar 2017.

[BH12] Nathan Bell and Jared Hoberock. Thrust: A productivity-oriented library for cuda. In *GPU computing gems Jade edition*, pages 359–371. Elsevier, 2012.

[BLK+12] Danielle D Brown, Scott LaPoint, Roland Kays, Wolfgang Heidrich, Franz Kümmeth, and Martin Wikelski. Accelerometer-informed gps

telemetry: Reducing the trade-off between resolution and longevity. *Wildlife Society Bulletin*, 36(1):139–146, 2012.

[BM13] Sean Baxter and Duane Merrill. Efficient merge, search, and set operations on gpus | gtc 2013. 2013.

[BND+20] Annika Bonerath, Benjamin Niedermann, Jim Diederich, Yannick Orgeig, Johannes Oehrlein, and Jan-Henrik Haunert. A Time-Windowed Data Structure for Spatial Density Maps. page 10, 2020.

[BOA09] Markus Billeter, Ola Olsson, and Ulf Assarsson. Efficient stream compaction on wide simd many-core architectures. In *Proceedings of the conference on high performance graphics 2009*, pages 159–166, 2009.

[BRPY+15a] Guillaume Bastille-Rousseau, Jonathan R. Potts, Charles B. Yackulic, Jacqueline L. Frair, E. Hance Ellington, and Stephen Blake. Flexible characterization of animal movement pattern using net squared displacement and a latent state model. *Movement Ecology*, 4(1):15, dec 2015.

[BRPY+15b] Guillaume Bastille-Rousseau, Jonathan R. Potts, Charles B. Yackulic, Jacqueline L. Frair, E. Hance Ellington, and Stephen Blake. Flexible characterization of animal movement pattern using net squared displacement and a latent state model. *Movement Ecology*, 4(1):15, dec 2015.

[BRPY+16] Guillaume Bastille-Rousseau, Jonathan R Potts, Charles B Yackulic, Jacqueline L Frair, E Hance Ellington, and Stephen Blake. Flexible characterization of animal movement pattern using net squared displacement and a latent state model. *Movement ecology*, 4(1):1–12, 2016.

[Cal06] C Calenge. The package adehabitat for the R software: tool for the analysis of space and habitat use by animals. *Ecological Modelling*, 197:1035, 2006.

[Cam05] Lisa M. Campbell. Overcoming Obstacles to Interdisciplinary Research. *Conservation Biology*, 19(2):574–577, apr 2005.

[ÇBAD17] Arzu Çöltekin, Susanne Bleisch, Gennady Andrienko, and Jason Dykes. Persistent challenges in geovisualization–a community perspective. *International Journal of Cartography*, 3(sup1):115–139, 2017.

[CBM+21] Zhanglong Cao, David Bryant, Timothy CA Molteno, Colin Fox, and Matthew Parry. V-spline: An adaptive smoothing spline for trajectory reconstruction. *Sensors*, 21(9):3215, 2021.

[CBT10] Edward A Codling, Rachel N Bearon, and Graeme J Thorn. Diffusion about the mean drift location in a biased random walk. *Ecology*, 91(10):3106–3113, 2010.

[CDRC09] Clément Calenge, Stéphane Dray, and Manuela Royer-Carenzi. The concept of animals' trajectories from a data analysis perspective. *Ecological informatics*, 4(1):34–41, 2009.

[CGL] Tony F Chan, Gene H Golub, and Randall J Leveque. UPDATING FORMULAE AND A PAIRWISE ALGORITHM FOR COMPUTING SAMPLE VARIANCES. Technical report.

[ÇGS+20] Arzu Çöltekin, Amy L. Griffin, Aidan Slingsby, Anthony C. Robinson, Sidonie Christophe, Victoria Rautenbach, Min Chen, Christopher Pettit, and Alexander Klippel. Geospatial Information Visualization and

Extended Reality Displays. In *Manual of Digital Earth*, pages 229–277. Springer Singapore, 2020.

[CJF18] Arzu Cöltekin, Halldór Janetzko, and Sara I Fabrikant. Geovisualization. *Zurich Open Repository and Archive*, page online, 2018.

[Cla89] David D. Clark. Overview of the ARGOS system. pages 934–939. Publ by IEEE, 1989.

[CLY17] Siming Chen, Lijing Lin, and Xiaoru Yuan. Social media visual analytics. In *Computer Graphics Forum*, volume 36, pages 563–587. Wiley Online Library, 2017.

[Cor] Omar Cornut. Github - ocornut/imgui: Dear imgui: Bloat-free graphical user interface for c++ with minimal dependencies.

[Cor01] John Corbett. Charles joseph minard, mapping napoleon's march, 1861. csiss classics. 2001.

[CP] M. Châteauneuf and Seine . Préfecture. Rapport sur la marche et les effets du choléra-morbus dans paris et les communes rurales du département de la seine. *undefined*.

[CPB08] Edward A Codling, Michael J Plank, and Simon Benhamou. Random walk models in biology. *Journal of the Royal society interface*, 5(25):813–834, 2008.

[CT05] KA Cook and JJ Thomas. Illuminating the path: The research and development agenda for visual analytics. 2005.

[CVB+20] G Cerritelli, L Vanni, NE Baldaccini, A Lenzoni, M Sorrenti, V Falchi, P Luschi, and D Giunchi. Simpler methods can outperform more sophisticated ones when assessing bird migration starting date. *Journal of Ornithology*, 161(3):901–907, 2020.

[DAA+19] Alexander Dunkel, Gennady Andrienko, Natalia Andrienko, Dirk Burghardt, Eva Hauthal, and Ross Purves. A conceptual framework for studying collective reactions to events in location-based social media. *International Journal of Geographical Information Science*, 33(4):780–804, 2019.

[DBC+15] Urška Demšar, Kevin Buchin, Francesca Cagnacci, Kamran Safi, Bettina Speckmann, Nico Van de Weghe, Daniel Weiskopf, and Robert Weibel. Analysis and visualisation of movement: an interdisciplinary review. *Movement Ecology*, 3(1):5, dec 2015.

[DBW+13] Somayeh Dodge, Gil Bohrer, Rolf Weinzierl, Sarah C. Davidson, Roland Kays, David Douglas, Sebastian Cruz, Jiawei Han, David Brandes, and Martin Wikelski. The environmental-data automated track annotation (env-data) system: linking animal tracks with environmental data. *Movement Ecology*, 1(1):3, Jul 2013.

[Dem08] Biadgilgn Demissie Mullaw. Moving Objects in Static Maps, Animation, and the Space-Time Cube. Technical report, 2008.

[DF70] Henry Clifford Darby and Harold Fullard. *The New Cambridge Modern History Atlas*. Cambridge University Press, 1970.

[Did15] K Didan. MOD13C2 MODIS/Terra Vegetation Indices Monthly L3 Global 0.05Deg CMG V006. Distributed by NASA EOSDIS Land Processes DAAC., 2015.

[DLBP+21] Urška Demšar, Jed A Long, Fernando Benitez-Paez, Vanessa Brum Bastos, Solène Marion, Gina Martin, Sebastijan Sekulić, Kamil Smolak, Beate Zein, and Katarzyna Siła-Nowicka. Establishing the integrated science of movement: bringing together concepts and methods from animal and human movement analysis. *International Journal of Geographical Information Science*, pages 1–36, 2021.

[DMÁM+19] Marcello D'Amico, Ricardo C Martins, Jose M Álvarez-Martínez, Miguel Porto, Rafael Barrientos, and Francisco Moreira. Bird collisions with power lines: Prioritizing species and areas by estimating potential population-level impacts. *Diversity and Distributions*, 25(6):975–982, 2019.

[DPS+10] José Duato, Antonio J Pena, Federico Silla, Rafael Mayo, and Enrique S Quintana-Ortí. rcuda: Reducing the number of gpu-based accelerators in high performance clusters. In *2010 International Conference on High Performance Computing & Simulation*, pages 224–231. IEEE, 2010.

[DSK05] Nicholas J DeCesare, John R Squires, and Jay A Kolbe. Effect of forest canopy on gps-based movement data. *Wildlife Society Bulletin*, 33(3):935–941, 2005.

[DSTE20] Sebastian Dunnett, Alessandro Sorichetta, Gail Taylor, and Felix Eigenbrod. Harmonised global datasets of wind and solar farm locations and power. *Scientific Data*, 7(1):130, 2020.

[DV10a] Urška Demšar and Kirsi Virrantaus. Space-time density of trajectories: Exploring spatio-temporal patterns in movement data. *International Journal of Geographical Information Science*, 24(10):1527–1542, 2010.

[DV10b] Urška Demšar and Kirsi Virrantaus. Space–time density of trajectories: exploring spatio-temporal patterns in movement data. *International Journal of Geographical Information Science*, 24(10):1527–1542, oct 2010.

[DWA+16] Somayeh Dodge, Robert Weibel, Sean C. Ahearn, Maike Buchin, and Jennifer A. Miller. Analysis of movement data. *International Journal of Geographical Information Science*, 30(5):825–834, may 2016.

[DWL08] Somayeh Dodge, Robert Weibel, and Anna-Katharina Lautenschütz. Towards a taxonomy of movement patterns. *Information visualization*, 7(3-4):240–252, 2008.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[ESB16] Hendrik Edelhoff, Johannes Signer, and Niko Balkenhol. Path segmentation for beginners: an overview of current methods for detecting changes in animal movement patterns. *Movement Ecology*, 4(1):21, dec 2016.

[Esr12] Esri. ArcGIS 10.1 projection engine tables: geographic and vertical coordinate systems, datums and vertical datums, spheroids, prime meridians, and angular units of measure. Technical report, 2012.

[FC17] Christen H Fleming and Justin M Calabrese. A new kernel density estimator for accurate home-range and species-range area estimation. *Methods in Ecology and Evolution*, 8(5):571–579, 2017.

[FCRA19] Aldina Franco, Ines Catry, Kate Rogerson, and Marta Acacio. White stork adults and juveniles 2016 - 2019, 2019.

[FFH+10] Jacqueline L Frair, John Fieberg, Mark Hebblewhite, Francesca Cagnacci, Nicholas J DeCesare, and Luca Pedrotti. Resolving issues of imprecise and habitat-biased locations in ecological analyses using gps telemetry data. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1550):2187–2200, 2010.

[FH15] Ruth Falconer and Alasdair Houston. Visual Simulation of Soil-Microbial System Using GPGPU Technology. *Computation*, 3(1):58–71, feb 2015.

[FKH] Alex Feurst, Charles Kazer, and William Hoffman. Qgis python plugins repository.

[FRC+07] Tom G. Farr, Paul A. Rosen, Edward Caro, Robert Crippen, Riley Duren, Scott Hensley, Michael Kobrick, Mimi Paller, Ernesto Rodriguez, Ladislav Roth, David Seal, Scott Shaffer, Joanne Shimada, Jeffrey Umland, Marian Werner, Michael Oskin, Douglas Burbank, and Douglas Alsdorf. The shuttle radar topography mission. *Rev. Geophys.*, 45:–, may 2007.

[FRW10] Youyi Fong, Håvard Rue, and Jon Wakefield. Bayesian inference for generalized linear mixed models. *Biostatistics*, 11(3):397–412, 2010.

[FSF+18] CH Fleming, D Sheldon, WF Fagan, P Leimgruber, T Mueller, D Nandintsetseg, MJ Noonan, KA Olson, Edy Setyawan, A Sianipar, et al. Correcting for missing and irregular data in home-range estimation. *Ecological Applications*, 28(4):1003–1010, 2018.

[FSS+17] Kevin C Fraser, Amanda Shave, Anne Savage, Alisha Ritchie, Kelsey Bell, Joseph Siegrist, James D Ray, Kelly Applegate, and Myrna Pearman. Determining fine-scale migratory connectivity and habitat selection for a migratory songbird by using new gps technology. *Journal of Avian Biology*, 48(3):339–345, 2017.

[FT03] Per Fauchald and Torkild Tveraa. Using first-passage time in the analysis of area-restricted search and habitat selection. *Ecology*, 84(2):282–288, 2003.

[GCS+16a] Nathalie I. Gilbert, Ricardo A. Correia, João Paulo Silva, Carlos Pacheco, Inês Catry, Philip W. Atkinson, Jenny A. Gill, and Aldina M. A. Franco. 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):7, dec 2016.

[GCS+16b] Nathalie I. Gilbert, Ricardo A. Correia, João Paulo Silva, Carlos Pacheco, Inês Catry, Philip W. Atkinson, Jenny A. Gill, and Aldina

M. A. Franco. 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):7, Mar 2016.

[GD20] Anita Graser and Melitta Dragaschnig. Open Geospatial Tools for Movement Data Exploration. 70:3–10, 2020.

[GG13] Evgeniy Yur Evich Gorodov and Vasiliy Vasil Evich Gubarev. Analytical review of data visualization methods in application to big data. *Journal of Electrical and Computer Engineering*, 2013.

[GH95] E Grafarend and A Heidenreich. The generalized mollweide projection of the biaxial ellipsoid. *Bulletin géodésique*, 69(3):164–172, 1995.

[GH15] Amir Gandomi and Murtaza Haider. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35:137–144, 4 2015.

[GKD10] Kate Haley Goldman, Cheryl Kessler, and Elizabeth Danter. Science on a sphere®. *Retrieved December*, 31:2018, 2010.

[GMB12] Oded Green, Robert McColl, and David A Bader. Gpu merge path: a gpu merging algorithm. In *Proceedings of the 26th ACM international conference on Supercomputing*, pages 331–340, 2012.

[Goo] Map and tile coordinates, maps javascript api, google developers. accessed: June 2021.

[Goo18] Michael F. Goodchild. Reimagining the history of GIS. *Annals of GIS*, 24(1):1–8, jan 2018.

[GS08] Wayne M Getz and David Saltz. A framework for generating and analyzing movement paths on ecological landscapes. *Proceedings of the National Academy of Sciences*, 105(49):19066–19071, 2008.

[GSA+ng] Jethro G. Gauld, João P Silva, Phillip W. Atkinson, Paul Record, Marta Acácio, Volen Arkumarev, Julio Blas, Willem Bouten, Niall Burton, Inês Catry, Jocelyn Champagnon, Gary D. Clewley, Olivier Dagys, Mindaugasand Duriez, Michael Exo, Wolfgang Fiedler, Andrea Flack, Guilad Friedemann, Johannes Fritz, Clara Garcia-Ripolles, Stefan Garthe, Dimitri Giunchi, Atanas Grozdanov, Roi Harel, Elizabeth M. Humphreys, René Janssen, Andrea Kölzsch, Olga Kulikova, Thomas K. Lameris, Pascual López-López, Elizabeth A. Masden, Flavio Monti, Ran Nathan, Stoyan Nikolov, Steffen Oppel, Hristo Peshev, Louis Phipps, Ivan Pokrovsky, Viola H. Ross-Smith, Victoria Saravia-Mullin, Emily S. Scragg, Andrea Sforzi, Emilian Stoynov, Chris Thaxter, Wouter Van Steelant, Mariëlle van Toor, Bernd Vorneweg, Jonas Waldenström, Martin Wikelski, Ramūnas Žydelis, and Aldina M. A. Franco. Hotspots in the grid: Avian sensitivity and vulnerability to collision risk from energy infrastructure interactions in europe and north africa. *Journal of Applied Ecology*, forthcoming.

[Gui13] Design Guide. Cuda c best practices guide. *NVIDIA, July*, 2013.

[GWD20] Anita Graser, Peter Widhalm, and Melitta Dragaschnig. The M³ massive movement model: a distributed incrementally updatable solution for big movement data exploration. *International Journal of Geographical Information Science*, 34(12):2517–2540, dec 2020.

[GWS11] Adrian C. Gleiss, Rory P. Wilson, and Emily L.C. Shepard. Making overall dynamic body acceleration work: On the theory of acceleration as a proxy for energy expenditure. *Methods in Ecology and Evolution*, 2(1):23–33, jan 2011.

[H+68] Torsten Hagerstrand et al. Innovation diffusion as a spatial process. *Innovation diffusion as a spatial process.*, 1968.

[Har12] Mark Harris. How to Overlap Data Transfers in CUDA C/C++ | NVIDIA Developer Blog, 2012.

[Har18] Jr Frank E Harrell. Hmisc Package, 2018.

[HFHM15] Hadley Wickham, Romain François, Lionel Henry, and Kirill Müller. DPLYR, 2015.

[HGP+18] Wenbin He, Hanqi Guo, Tom Peterka, Sheng Di, Franck Cappello, and Han-Wei Shen. Parallel partial reduction for large-scale data analysis and visualization. In *2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)*, pages 45–55. IEEE, 2018.

[HHH+19] M. P.G. Hofman, M. W. Hayward, M. Heim, P. Marchand, C. M. Rolandsen, J. Mattisson, F. Urbano, M. Heurich, A. Mysterud, J. Melzheimer, N. Morellet, U. Voigt, B. L. Allen, B. Gehr, C. Rouco, W. Ullmann, Holand, N. H. Jørgensen, G. Steinheim, F. Cagnacci, M. Kroeschel, P. Kaczensky, B. Buuveibaatar, J. C. Payne, I. Palmegiani, K. Jerina, P. Kjellander, Johansson, S. LaPoint, R. Bayrakcismith, J. D.C. Linnell, M. Zaccaroni, M. L.S. Jorge, J. E.F. Oshima, A. Songhurst, C. Fischer, R. T. Mc Bride, J. J. Thompson,

S. Streif, R. Sandfort, C. Bonenfant, M. Drouilly, M. Klapproth, D. Zinner, R. Yarnell, A. Stronza, L. Wilmott, E. Meisingset, M. Thaker, A. T. Vanak, S. Nicoloso, R. Graeber, S. Said, M. R. Boudreau, A. Devlin, R. Hoogesteijn, J. A. May-Junior, J. C. Nifong, J. Odden, H. B. Quigley, F. Tortato, D. M. Parker, A. Caso, J. Perrine, C. Tellaeche, F. Zieba, T. Zwijacz-Kozica, C. L. Appel, I. Axsom, W. T. Bean, B. Cristescu, S. Périquet, K. J. Teichman, S. Karpanty, A. Licoppe, V. Menges, K. Black, T. L. Scheppers, S. C. Schai-Braun, F. C. Azevedo, F. G. Lemos, A. Payne, L. H. Swanepoel, B. V. Weckworth, A. Berger, A. Bertassoni, G. McCulloch, P. Šustr, V. Athreya, D. Bockmuhl, J. Casaer, A. Ekori, D. Melovski, C. Richard-Hansen, D. Van De Vyver, R. Reyna-Hurtado, E. Robardet, N. Selva, A. Sergiel, M. S. Farhadinia, P. Sunde, R. Portas, H. Ambarli, R. Berzins, P. M. Kappeler, G. K. Mann, L. Pyritz, C. Bissett, T. Grant, R. Steinmetz, L. Swedell, R. J. Welch, D. Armenteras, O. R. Bidder, T. M. González, A. Rosenblatt, S. Kachel, and N. Balkenhol. Right on track? Performance of satellite telemetry in terrestrial wildlife research. *PLoS ONE*, 14(5):1–26, 2019.

[HHKP19] Sounman Hong, Sun Hyoung Kim, Youngrok Kim, and Jeongin Park. Big Data and government: Evidence of the role of Big Data for smart cities. *Big Data and Society*, 6(1), jan 2019.

[HLG+] Mark Jason Harris, Anselmo Lastra, Advisor Gary Bishop, Ming C Lin, David Leith, and Frederick P Brooks. Real-Time Cloud Simulation and Rendering. Technical report.

[HLZL20] Yu Huang, Yan Li, Zhaofeng Zhang, and Ryan Wen Liu. GPU-Accelerated Compression and Visualization of Large-Scale Vessel Trajectories in Maritime IoT Industries. *IEEE Internet of Things Journal*, 7(11):10794–10812, nov 2020.

[HPB10] Niels T Hintzen, Gerjan J Piet, and Thomas Brunel. Improved estimation of trawling tracks using cubic hermite spline interpolation of position registration data. *Fisheries Research*, 101(1-2):108–115, 2010.

[HR08] Michael C Hansen and Robert A Riggs. Accuracy, precision, and observation rates of global positioning system telemetry collars. *The Journal of Wildlife Management*, 72(2):518–526, 2008.

[HRBP15] Tomislav Hengl, Pierre Roudier, Dylan Beaudette, and Edzer Pebesma. Plotkml: Scientific visualization of spatio-temporal data. *Journal of Statistical Software*, 63(5):1–25, jan 2015.

[HSC19] Koen Hufkens, Reto Stauffer, and Elio Campitelli. The ecwmfr package: an interface to ecmwf api endpoints, 2019.

[HSLS20] Maurice Herlihy, Nir Shavit, Victor Luchangco, and Michael Spear. *The art of multiprocessor programming*. Newnes, 2020.

[HvE12] Robert J. Hijmans and Jacob van Etten. *raster: Geographic analysis and modeling with raster data*, 2012. R package version 2.0-12.

[HWV15] R. J. Hijmans, E. Williams, and C. Vennes. Geosphere: spherical trigonometry, 2015.

[IKJM05] James W Cain III, Paul R Krausman, Brian D Jansen, and John R Morgart. Influence of topography and gps fix interval on gps collar performance. *Wildlife Society Bulletin*, 33(3):926–934, 2005.

[JBC+19] Rocio Joo, Matthew E. Boone, Thomas A. Clay, Samantha C Patrick, Susana Clusella-Trullas, and Mathieu Basille. Navigating through the R packages for movement. *Journal of Animal Ecology*, 2019.

[Jen12] Bernhard Jenny. Adaptive composite map projections. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2575–2582, 2012.

[JLLD08] Devin S Johnson, Joshua M London, Mary-Anne Lea, and John W Durban. Continuous-time correlated random walk model for animal telemetry data. *Ecology*, 89(5):1208–1215, 2008.

[JSM+18] Bernhard Jenny, Daniel M Stephen, Ian Muehlenhaus, Brooke E Marston, Ritesh Sharma, Eugene Zhang, and Helen Jenny. Design principles for origin-destination flow maps. *Cartography and Geographic Information Science*, 45(1):62–75, 2018.

[KBMK+19] Joseph Kiesecker, Sharon Baruch-Mordo, Christina M. Kennedy, James R. Oakleaf, Alessandro Baccini, and Bronson W. Griscom. Hitting the Target but Missing the Mark: Unintended Environmental Consequences of the Paris Climate Agreement. *Frontiers in Environmental Science*, 7(October), 2019.

[KCN15] Cheikh Kacfah Emani, Nadine Cullot, and Christophe Nicolle. Understandable Big Data: A survey, aug 2015.

[Ket14] Simon Kettle. Geodesic distances: How long is that line again? - esri community, 2014.

[Ket17] Simon Kettle. Distance on an ellipsoid: Vincenty's formulae - esri community, 2017.

[KHKC+18] JC Kleyheeg-Hartman, KL Krijgsveld, MP Collier, MJM Poot, AR Boon, TA Troost, and S Dirksen. Predicting bird collisions with wind turbines: Comparison of the new empirical flux collision model with the soss band model. *Ecological Modelling*, 387:144–153, 2018.

[Kir07] David Kirk. NVIDIA CUDA software and GPU parallel computing architecture. In *International Symposium on Memory Management, ISMM*, page 103, 2007.

[KK92] A. Koussoulakou and M.J. Kraak. Spatia-temporal maps and cartographic communication. *The Cartographic Journal*, 29:101–108, 12 1992.

[KO20] Menno-Jan Kraak and Ferjan Ormeling. *Cartography: visualization of geospatial data.* 2020.

[Koo65] Gerald L Kooyman. Techniques used in measuring diving capacities of weddell seals. *Polar Record*, 12(79):391–394, 1965.

[Kra03a] Menno Jan Kraak. Geovisualization illustrated. In *ISPRS Journal of Photogrammetry and Remote Sensing*, volume 57, pages 390–399. Elsevier, apr 2003.

[Kra03b] Menno-Jan Kraak. The space-time cube revisited from a geovisualization perspective. In *Proc. 21st International Cartographic Conference*, pages 1988–1996, 2003.

[KWZ19] Mahmoud Khairy, Amr G. Wassal, and Mohamed Zahran. A survey of architectural approaches for improving GPGPU performance, programmability and heterogeneity. *Journal of Parallel and Distributed Computing*, 127:65–88, may 2019.

[LCDC14] Mael Le Corre, Christian Dussault, and Steeve D Côté. Detecting changes in the annual movements of terrestrial migratory species: using the first-passage time to document the spring migration of caribou. *Movement ecology*, 2(1):1–11, 2014.

[LGWK13] Scott LaPoint, Paul Gallery, Martin Wikelski, and Roland Kays. Animal behavior, cost-based corridor models, and real corridors. *Landscape Ecology*, 28(8):1615–1630, oct 2013.

[LHW20] Wenwei Liao, Eric J Hackathorn, and Suefen Wang. A study of applying a 3d instructional software of popular science (science on a sphere explorer) in earth science education of an elementary school. In *2020 22nd International Conference on Advanced Communication Technology (ICACT)*, pages 503–506. IEEE, 2020.

[LK15] Jae Gil Lee and Minseo Kang. Geospatial Big Data: Challenges and Opportunities. *Big Data Research*, 2(2):74–81, jun 2015.

[LLX19] Zhuohang Lai, Qiong Luo, and Xiaolong Xie. Efficient data-parallel primitives on heterogeneous systems. In *Proceedings of the 48th International Conference on Parallel Processing*, pages 1–10, 2019.

[LNOM08] Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. Nvidia tesla: A unified graphics and computing architecture. *IEEE micro*, 28(2):39–55, 2008.

[LO01] Doug Laney and Others. 3D data management: Controlling data volume, velocity and variety. *META group research note*, 6(70):1, 2001.

[Lon16] Jed A. Long. Kinematic interpolation of movement data. *International Journal of Geographical Information Science*, 30(5):854–868, may 2016.

[LRGV07] Jesse S Lewis, Janet L Rachlow, Edward O Garton, and Lee A Vierling. Effects of habitat on gps collar performance: using data screening to reduce location error. *Journal of applied ecology*, 44(3):663–671, 2007.

[LS73] Peter R Lipow and IJ Schoenberg. Cardinal interpolation and spline functions. iii. cardinal hermite interpolation. *Linear Algebra and its Applications*, 6:273–304, 1973.

[Mac94] Alan M. Maceachren. Visualization in modern cartography: Setting the agenda. *Modern Cartography Series*, 2:1–12, 1 1994.

[MCNF92] B. J. McConnell, C. Chambers, K. S. Nicholas, and M. A. Fedak. Satellite tracking of grey seals (Halichoerus grypus). *Journal of Zoology*, 226(2):271–282, 1992.

[ME02] Juan Manuel Morales and Stephen P Ellner. Scaling up animal movements in heterogeneous landscapes: the importance of behavior. *Ecology*, 83(8):2240–2247, 2002.

[MGM+11] Aaftab Munshi, Benedict Gaster, Timothy G. Mattson, James Fung, and Dan Ginsburg. *OpenCL Programming Guide*. Addison-Wesley Professional, 1st edition, 2011.

[MGP+04] Alan M MacEachren, Mark Gahegan, William Pike, Isaac Brewer, Guoray Cai, Eugene Lengerich, and F Hardistry. Geovisualization for knowledge construction and decision support. *IEEE computer graphics and applications*, 24(1):13–17, 2004.

[MGS+18] Flavio Monti, David Grémillet, Andrea Sforzi, Giampiero Sammuri, Jean Marie Dominici, Rafel Triay Bagur, Antoni Muñoz Navarro, Leonida Fusani, and Olivier Duriez. Migration and wintering strategies in vulnerable mediterranean osprey populations. *Ibis*, 160(3):554–567, 2018.

[MJLW02] David P Maitland, Rebecca L Jackson, Richard J Ladle, and Paul Ward. Field considerations and problems associated with radio tracking a tropical fresh-water land crab. *Journal of Crustacean Biology*, 22(2):493–496, 2002.

[MLM09] Hannah W Mckenzie, Mark A Lewis, and Evelyn H Merrill. First Passage Time Analysis of Animal Movement and Insights into the Functional Response. *Bulletin of Mathematical Biology*, 71:107–129, 2009.

[MNU00] Andrew McCallum, Kamal Nigam, and Lyle H Ungar. 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*, pages 169–178, 2000.

[MP19] Roxana Mihet and Thomas Philippon. The economics of big data and artificial intelligence. In *International Finance Review*, volume 20, pages 29–43. Emerald Group Publishing Ltd., oct 2019.

[Mro18] Tomasz Mrozewski. Geospatial data and software reviews: Movebank, dec 2018.

[MS18] Donald James McLean and Marta A. Skowron Volponi. <tt>trajr</tt> : An R package for characterisation of animal trajectories. *Ethology*, 124(6):440–448, jun 2018.

[MS19] Joaquín Muñoz-Sabater. Era5-land hourly data from 1981 to present. copernicus climate change service (c3s) climate data store (cds)., 2019.

[MSH+20] Ana T. Marques, Carlos D. Santos, Frank Hanssen, Antonio Román Muñoz, Alejandro Onrubia, Martin Wikelski, Francisco Moreira, Jorge Manuel Palmeirim, and João P. Silva. Wind turbines cause functional habitat loss for migratory soaring birds. *Journal of Animal Ecology*, 89(1):93–103, 2020.

[Mun09] Aaftab Munshi. The opencl specification. In *2009 IEEE Hot Chips 21 Symposium (HCS)*, pages 1–314. IEEE, 2009.

[Nat19] National Science Foundation. Open Topography Data Portal, 2019.

[NCTB17] Marco S. Nobile, Paolo Cazzaniga, Andrea Tangherloni, and Daniela Besozzi. Graphics processing units in bioinformatics, computational biology and systems biology. *Briefings in Bioinformatics*, 18(5):870–885, sep 2017.

[NGR+08] Ran Nathan, Wayne M Getz, Eloy Revilla, Marcel Holyoak, Ronen Kadmon, David Saltz, and Peter E Smouse. A movement ecology paradigm for unifying organismal movement research. *Proceedings of the National Academy of Sciences of the United States of America*, 105(49):19052–9, dec 2008.

[NLLS15] Trisalyn Nelson, Jed Long, Karen Laberee, and Benjamin Stewart. A time geographic approach for delineating areas of sustained wildlife use. *Annals of GIS*, 21(1):81–90, 2015.

[NM13] Markus Neteler and Helena Mitasova. *Open source GIS: a GRASS GIS approach*, volume 689. Springer Science & Business Media, 2013.

[NN00] NGA and NASA. Shuttle Radar Topography Mission (SRTM GL1) Global 30m Ellipsoidal, 2000.

[NSFR+12] R. Nathan, O. Spiegel, S. Fortmann-Roe, R. Harel, M. Wikelski, and W. M. Getz. Using tri-axial acceleration data to identify behavioral modes of free-ranging animals: general concepts and tools illustrated for griffon vultures. *Journal of Experimental Biology*, 215(6):986–996, 2012.

[NT20] Behnam Nikparvar and Jean-Claude Thill. A Density-Based Measure of Port Seaside Space-Time Utilization. pages 55–70. Springer, Cham, 2020.

[OA18] Kenichi W. Okamoto and Priyanga Amarasekare. A framework for high-throughput eco-evolutionary simulations integrating multilocus forward-time population genetics and community ecology. *Methods in Ecology and Evolution*, 9(3):525–534, mar 2018.

[ORJ15] Ryszard Oleksy, Paul A Racey, and Gareth Jones. High-resolution gps tracking reveals habitat selection and the potential for long-distance seed dispersal by madagascan flying foxes pteropus rufus. *Global Ecology and Conservation*, 3:678–692, 2015.

[OSFM10] N Owen-Smith, JM Fryxell, and EH Merrill. Foraging theory upscaled: the behavioural ecology of herbivore movement. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1550):2267–2278, 2010.

[PBBG09] Toby A Patterson, Marinelle Basson, Mark V Bravington, and John S Gunn. Classifying movement behaviour in relation to environmental conditions using hidden markov models. *Journal of Animal Ecology*, 78(6):1113–1123, 2009.

[PBS+18] Jonathan R. Potts, Luca Börger, D. Michael Scantlebury, Nigel C. Bennett, Abdulaziz Alagaili, and Rory P. Wilson. Finding turning-points in ultra-high-resolution animal movement data. *Methods in Ecology and Evolution*, 9(10):2091–2101, oct 2018.

[PCD+20] Guillaume Péron, Justin M Calabrese, Olivier Duriez, Christen H Fleming, Ruth García-Jiménez, Alison Johnston, Sergio Lambertucci, Kamran Safi, and Emily L C Shepard. The challenges of estimating the distribution of flight heights from telemetry or altimetry data. *Animal Biotelemetry*, 8(5):1 – 13, 2020.

[PDGM+21] Cristian Pasquaretta, Thibault Dubois, Tamara Gomez-Moracho, Virginie P Delepoulle, Guillaume Le Loc'h, Philipp Heeb, and Mathieu

Lihoreau. Analysis of temporal patterns in animal movement networks. *Methods in Ecology and Evolution*, 12(1):101–113, 2021.

[PDH+18] Sharon A. Poessel, Adam E. Duerr, Jonathan C. Hall, Melissa A. Braham, and Todd E. Katzner. Improving estimation of flight altitude in wildlife telemetry studies. *Journal of Applied Ecology*, 55(4):2064–2070, 2018.

[PFD+17] Guillaume Péron, Christen H. Fleming, Olivier Duriez, Julie Fluhr, Christian Itty, Sergio Lambertucci, Kamran Safi, Emily L. C. Shepard, and Justin M. Calabrese. The energy landscape predicts flight height and wind turbine collision hazard in three species of large soaring raptor. *Journal of Applied Ecology*, 54(6):1895–1906, 2017.

[PFG+21] Jessica Ann Phillips, Annette L Fayet, Tim Guilford, Fabrizio Manco, Victoria Warwick-Evans, and Phil Trathan. Foraging conditions for breeding penguins improve with distance from colony and progression of the breeding season at the south orkney islands. *Movement ecology*, 9(1):1–14, 2021.

[PM15] Mrinal Patwardhan and Sahana Murthy. When does higher degree of interaction lead to higher learning in visualizations? exploring the role of 'interactivity enriching features'. *Computers & Education*, 82:292–305, 2015.

[Pri92] I Priede. Wildlife telemetry: an introduction. *Wildlife Telemetry: Remote Monitoring and Tracking of Animals*, jan 1992.

[QCW+12] Lama Qasem, Antonia Cardew, Alexis Wilson, Iwan Griffiths, Lewis G. Halsey, Emily L.C. Shepard, Adrian C. Gleiss, and Rory Wilson. Tri-axial dynamic acceleration as a proxy for animal energy expenditure; should we be summing values or calculating the vector? *PLoS ONE*, 7(2):e31187, feb 2012.

[QGI19] QGIS Development Team. QGIS Geographic Information System, 2019.

[Rap93] Richard H Rapp. GEOMETRIC GEODESY PART II. Technical report, 1993.

[RCS+19] David Severin Ryberg, Dilara Gulcin Caglayan, Sabrina Schmitt, Jochen Linßen, Detlef Stolten, and Martin Robinius. The future of European onshore wind energy potential: Detailed distribution and simulation of advanced turbine designs. *Energy*, 182:1222–1238, 2019.

[Ren18] Sandra Rendgen. *The Minard System: The Complete Statistical Graphics of Charles-Joseph Minard*. Chronicle Books, 2018.

[RH09] Christian Rutz and Graeme C. Hays. New frontiers in biologging science. *Biology Letters*, 5:289–292, 6 2009.

[RKT+18] Shay Rotics, Michael Kaatz, Sondra Turjeman, Damaris Zurell, Martin Wikelski, Nir Sapir, Ute Eggers, Wolfgang Fiedler, Florian Jeltsch, and Ran Nathan. Early arrival at breeding grounds: Causes, costs and a trade-off with overwintering latitude. *Journal of Animal Ecology*, 87(6):1627–1638, 2018.

[RMAL18] Richard C Roberts, Liam Mcnabb, Naif Alharbi, and Robert S Laramee. Spectrum: A C++ Header Library for Colour Map Management. Technical report, 2018.

[RMAM14] N. Rahemi, M. R. Mosavi, A. A. Abedi, and S. Mirzakuchaki. Accurate Solution of Navigation Equations in GPS Receivers for Very High Velocities Using Pseudorange Measurements. *Advances in Aerospace Engineering*, 2014:1–8, jun 2014.

[Rob17] Anthony Robinson. Geovisual Analytics. *Geographic Information Science & Technology Body of Knowledge*, 2017(Q3), jul 2017.

[Rod01] Arthur R. Rodgers. Recent Telemetry Technology. In *Radio Tracking and Animal Populations*, pages 79–121. Elsevier, jan 2001.

[RR97] Robert S Rempel and Arthur R Rodgers. Effects of differential correction on accuracy of a gps animal location system. *The Journal of Wildlife Management*, pages 525–530, 1997.

[RRFH20] B. A. Ricker, P. R. Rickles, G. A. Fagg, and M. E. Haklay. Tool, toolmaker, and scientist: case study experiences using GIS in interdisciplinary research. *https://doi.org/10.1080/15230406.2020.1748113*, 47(4):350–366, jul 2020.

[RT04] Rolf Rabenseifner and Jesper Larsson Träff. More efficient reduction algorithms for non-power-of-two number of processors in message-passing parallel systems. In *European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting*, pages 36–46. Springer, 2004.

[RWER05] Arthur H Robinson, David Woodward, Matthew H Edney, and Arthur Robinson. Putting "cartography" into the history of cartography: Arthur h. robinson, david woodward, and the creation of a discipline. *cartographicperspectives.org*, 2005.

[SAE16] Navinder J Singh, Andrew M Allen, and Göran Ericsson. Quantifying migration behaviour using net squared displacement approach: clarifications and caveats. *PLoS One*, 11(3):e0149594, 2016.

[SAF+20] Andrea Soriano-Redondo, Marta Acácio, Aldina M. A. Franco, Bruno Herlander Martins, Francisco Moreira, Katharine Rogerson, and Inês Catry. Testing alternative methods for estimation of bird migration phenology from GPS tracking data. *Ibis*, 162(2):581–588, apr 2020.

[San02] Eunice E Santos. Optimal and efficient algorithms for summing and prefix summing on parallel machines. *Journal of Parallel and Distributed Computing*, 62(4):517–543, 2002.

[SBvLP+12] Judy Shamoun-Baranes, E. Emiel van Loon, Ross S. Purves, Bettina Speckmann, Daniel Weiskopf, and C. J. Camphuysen. Analysis and visualization of animal movement. *Biology Letters*, 8(1):6–9, feb 2012.

[SBWW16] Sherub Sherub, Gil Bohrer, Martin Wikelski, and Rolf Weinzierl. Behavioural adaptations to flight into thin air. *Biology letters*, 12(10):20160432, 2016.

[SDCG18] Dana Paige Seidel, Eric Dougherty, Colin Carlson, and Wayne M. Getz. Ecological metrics and methods for GPS movement data, nov 2018.

[SDL] Krzysztof Sopyy, Pawee L Drozda, and Przemyss Law Górecki. LNAI 7267 - SVM with CUDA Accelerated Kernels for Big Sparse Problems.

[SFA19] Johannes Signer, John Fieberg, and Tal Avgar. Animal movement tools (amt): R package for managing tracking data and conducting habitat selection analyses. *Ecology and Evolution*, 9(2):880–890, jan 2019.

[SHG+08] Shubhabrata Sengupta, Mark Harris, Michael Garland, et al. Efficient parallel scan algorithms for gpus. *NVIDIA, Santa Clara, CA, Tech. Rep. NVR-2008-003*, 1(1):1–17, 2008.

[SHZO07] Shubhabrata Sengupta, Mark Harris, Yao Zhang, and John D Owens. Scan primitives for gpu computing. 2007.

[Sie84] Ewa Siekierska. Part 3: The merger of computer data and thematic mapping: Towards an electronic Atlas. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 21(2-3):110–120, 1984.

[Sil86] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[Sin84] Roger W Sinnott. Virtues of the Haversine. *S&T*, 68(2):158, 1984.

[SJ10] Lauren M Scott and Mark V Janikas. Spatial statistics in arcgis. In *Handbook of applied spatial analysis*, pages 27–41. Springer, 2010.

[SJWS15] Bojan Savric, Bernhard Jenny, Denis White, and Daniel R. Strebe. User preferences for world map projections. *Cartography and Geographic Information Science*, 42(5):398–409, oct 2015.

[SKSŚ17] Szymon Szewrański, Jan Kazak, Marta Sylla, and Małgorzata Świader. Spatial data analysis with the use of ArcGIS and Tableau systems. In *Lecture Notes in Geoinformation and Cartography*, number 9783319451220, pages 337–349. Springer Berlin Heidelberg, 2017.

[SM98] James A. Slater and Stephen Malys. WGS 84 — Past, Present and Future. pages 1–7. Springer, Berlin, Heidelberg, 1998.

[Sny87] John Parr Snyder. *Map projections–A working manual*, volume 1395. US Government Printing Office, 1987.

[Sou11] Andy South. rworldmap: A New R package for Mapping Global Data. Technical Report 1, 2011.

[SPFCC17] Ariana Strandburg-Peshkin, Damien R Farine, Margaret C Crofoot, and Iain D Couzin. Habitat and social factors shape individual decisions and emergent group structure during baboon collective movement. *Elife*, 6:e19505, 2017.

[ŠPJ19] Bojan Šavrič, Tom Patterson, and Bernhard Jenny. The Equal Earth map projection. *International Journal of Geographical Information Science*, 33(3):454–465, mar 2019.

[(SR13] NASA Shuttle Radar Topography Mission (SRTM). Shuttle radar topography mission (srtm) global. distributed by open topography., 2013.

[SRSW20] Jakob Schwalb-Willmann, Ruben Remelgado, Kamran Safi, and Martin Wegmann. movevis: Animating movement trajectories in synchronicity with static or temporally dynamic environmental data in r. *Methods in Ecology and Evolution*, 11(5):664–669, may 2020.

[SS13] Seref Sagiroglu and Duygu Sinanc. Big data: A review. In *Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, CTS 2013*, pages 42–47, 2013.

[SW19] Jakob Schwalb-Willmann. *moveVis: Movement Data Visualization*, 2019. R package version 0.10.1.

[SWQ+08] Emily LC Shepard, Rory P Wilson, Flavio Quintana, Agustina Gómez Laich, Nikolai Liebsch, Diego A Albareda, Lewis G Halsey, Adrian Gleiss, David T Morgan, Andrew E Myers, et al. Identification of animal movement patterns using tri-axial accelerometry. *Endangered Species Research*, 10(1):47–60, mar 2008.

[SWS+19] Seung Youb Ssin, James A Walsh, Ross T Smith, Andrew Cunningham, and Bruce H Thomas. Geogate: Correlating geo-temporal datasets using an augmented reality space-time cube and tangible interactions. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 210–219. IEEE, 2019.

[Szu16] Jakub Szuppe. Boost. compute: A parallel computing library for c++ based on opencl. In *Proceedings of the 4th International Workshop on OpenCL*, pages 1–39, 2016.

[TFKB10] Stanley M Tomkiewicz, Mark R Fuller, John G Kie, and Kirk K Bates. Global positioning system and associated technologies in animal behaviour and ecological research. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1550):2163–2176, 2010.

[TOSW15] Georgios Technitis, Walied Othman, Kamran Safi, and Robert Weibel. From A to B, randomly: a point-to-point random trajectory generator for animal movement. *International Journal of Geographical Information Science*, 29(6):912–934, jun 2015.

[TRK+18] Hannu Tikkanen, Seppo Rytkönen, Olli Pekka Karlin, Tuomo Ollila, Veli Matti Pakanen, Heikki Tuohimaa, and Markku Orell. Modelling golden eagle habitat selection and flight activity in their home ranges for

safer wind farm planning. *Environmental Impact Assessment Review*, 71(April):120–131, 2018.

[TSF+06] Yann Tremblay, Scott A. Shaffer, Shannon L. Fowler, Carey E. Kuhn, Birgitte I. McDonald, Michael J. Weise, Charle André Bost, Henri Weimerskirch, Daniel E. Crocker, Michael E. Goebel, and Daniel P. Costa. Interpolation of animal tracking data in a fluid environment. *Journal of Experimental Biology*, 209(1):128–140, jan 2006.

[Tuf85] Edward R Tufte. The visual display of quantitative information. *The Journal for Healthcare Quality (JHQ)*, 7(3):15, 1985.

[Tul18] Theodore H. Tulchinsky. John snow, cholera, the broad street pump; waterborne diseases then and now. *Case Studies in Public Health*, page 77, 2018.

[UCD+10] Ferdinando Urbano, Alison Cameron, Holger Dettki, Markus Neteler, Francesca Cagnacci, and Clément Calenge. Wildlife tracking data management: a new vision. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1550):2177–2185, jul 2010.

[Vin75] Thaddeus Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review*, 23(176):88–93, 1975.

[Vis10] Todd Vision. The Dryad Digital Repository: Published evolutionary data as part of the greater data ecosystem. *Nature Precedings*, pages 1–1, jun 2010.

[vO11] Jeremiah van Oosten. Cuda memory model, 2011.

[War08] Frank Warmerdam. The geospatial data abstraction library. In *Open source approaches in spatial data handling*, pages 87–104. Springer, 2008.

[WG12] Gary C White and Robert A Garrott. *Analysis of wildlife radio-tracking data*. Elsevier, 2012.

[WHR+21] Brian M Wood, Jacob A Harris, David A Raichlen, Herman Pontzer, Katherine Sayre, Amelia Sancilio, Colette Berbesque, Alyssa N Crittenden, Audax Mabulla, Richard McElreath, et al. Gendered movement ecology and landscape use in hadza hunter-gatherers. *Nature human behaviour*, 5(4):436–446, 2021.

[Wic16] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

[WK13] Changjun Wu and Ananth Kalyanaraman. GPU-Accelerated protein family identification for metagenomics. In *Proceedings - IEEE 27th International Parallel and Distributed Processing Symposium Workshops and PhD Forum, IPDPSW 2013*, pages 559–568. IEEE Computer Society, 2013.

[WLB+20] Quinn M R Webber, Michel P Laforge, Maegwin Bonar, Alec L Robitaille, Christopher Hart, Sana Zabihi-Seissan, and Eric Vander Wal. The ecology of individual differences empirically applied to space-use and movement tactics. *The American Naturalist*, 196(1):E1–E15, 2020.

[WM19] Kays R Wikelski M. Movebank: archive, analysis and sharing of animal movement data. Hosted by the Max Planck Institute for Animal Behavior., 2019.

[WNB+15] Christopher C. Wilmers, Barry Nickel, Caleb M. Bryce, Justine A. Smith, Rachel E. Wheat, Veronica Yovovich, and M. Hebblewhite. The golden age of bio-logging: How animal-borne sensors are advancing the frontiers of ecology. *Ecology*, 96(7):1741–1753, jul 2015.

[Woj20] Susan Wojcicki. YouTube at 15: My personal journey and the road ahead, feb 2020.

[WRM12] Robert F. Wong, Craig M. Rollins, and Clifton F. Minter. Recent Updates to the WGS 84 Reference Frame, sep 2012.

[WSN19] Jorge A Wagner Filho, Wolfgang Stuerzlinger, and Luciana Nedel. Evaluating an Immersive Space-Time Cube Geovisualization for Intuitive Trajectory Data Exploration. 2019.

[WTB+20] Hannah J. Williams, Lucy A. Taylor, Simon Benhamou, Allert I. Bijleveld, Thomas A. Clay, Sophie de Grissac, Urška Demšar, Holly M. English, Novella Franconi, Agustina Gómez-Laich, Rachael C. Griffiths, William P. Kay, Juan Manuel Morales, Jonathan R. Potts, Katharine F. Rogerson, Christian Rutz, Anouk Spelt, Alice M. Trevail, Rory P. Wilson, and Luca Börger. Optimizing the use of biologgers for movement ecology research, jan 2020.

[WWQ+06] Rory P. Wilson, Craig R. White, Flavio Quintana, Lewis G. Halsey, Nikolai Liebsch, Graham R. Martin, and Patrick J. Butler. Moving towards acceleration for estimates of activity-specific metabolic rate in free-living animals: The case of the cormorant. *Journal of Animal Ecology*, 75(5):1081–1090, sep 2006.

[WZW19] Shaohua Wang, Yang Zhong, and Erqi Wang. An integrated GIS platform architecture for spatiotemporal big data. *Future Generation Computer Systems*, 94:160–172, may 2019.

[XD15] Glenn Xavier and Somayeh Dodge. An exploratory visualization tool for mapping the relationships between animal movement and the environment. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Interacting with Maps - MapInteract '14*, pages 36–42, New York, New York, USA, 2015. ACM Press.

[Xie13] Yihui Xie. Animation: An R package for creating animations and demonstrating statistical methods. *Journal of Statistical Software*, 53(1):1–27, apr 2013.

[YDJ+19] Yalong Yang, Tim Dwyer, Bernhard Jenny, Kim Marriott, Maxime Cordeil, and Haohui Chen. Origin-Destination Flow Maps in Immersive Environments. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):693–703, jan 2019.

[YSK11] Cem Yuksel, Scott Schaefer, and John Keyser. Parameterization and applications of catmull–rom curves. *Computer-Aided Design*, 43(7):747–755, 2011.

[ZLB99] Yifeng Zhou, Henry Leung, and Martin Blanchette. Sensor alignment with earth-centered earth-fixed (ECEF) coordinate system. *IEEE Transactions on Aerospace and Electronic Systems*, 35(2):410–418, 1999.

[ZW15] G Hulley Z Wan, S Hook. MOD11C3 MODIS/Terra Land Surface Temperature/Emissivity Monthly L3 Global 0.05Deg CMG V006. Distributed by NASA EOSDIS Land Processes DAAC., 2015.

[ZWZM12] Chen Zhong, Tao Wang, Wei Zeng, and Stefan Müller Arisona. Spatiotemporal visualisation: A survey and outlook. *Communications in Computer and Information Science*, 242 COMMUN:299–317, 2012.

[ZZH17] Guiming Zhang, A-Xing Zhu, and Qunying Huang. A gpu-accelerated adaptive kernel density estimation approach for efficient point pattern analysis on spatial big data. *International Journal of Geographical Information Science*, 31(10):2068–2097, 2017.