FACULTY OF DIGITAL ENGINEERING
Computer Graphics Systems Group

A Software Framework for GPU-based Geo-Temporal Visualization Techniques

Dissertation
in partial fulfillment for the academic degree
"doctor rerum naturalium"
(Dr. rer. nat.)
in IT Systems Engineering

Hasso Plattner Institute | Faculty of Digital Engineering
University of Potsdam

submitted by
Stefan Buschmann

Supervision:
Prof. Dr. Jürgen Döllner

Potsdam,
November 15, 2018

Published online at the
Institutional Repository of the University of Potsdam:
https://doi.org/10.25932/publishup-44340
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-443406

Abstract

Spatio-temporal data denotes a category of data that contains spatial as well as temporal components. For example, time-series of geo-data, thematic maps that change over time, or tracking data of moving entities can be interpreted as spatio-temporal data.

In today's automated world, an increasing number of data sources exist, which constantly generate spatio-temporal data. This includes for example traffic surveillance systems, which gather movement data about human or vehicle movements, remote-sensing systems, which frequently scan our surroundings and produce digital representations of cities and landscapes, as well as sensor networks in different domains, such as logistics, animal behavior study, or climate research.

For the analysis of spatio-temporal data, in addition to automatic statistical and data mining methods, exploratory analysis methods are employed, which are based on interactive visualization. These analysis methods let users explore a data set by interactively manipulating a visualization, thereby employing the human cognitive system and knowledge of the users to find patterns and gain insight into the data.

This thesis describes a software framework for the visualization of spatio-temporal data, which consists of GPU-based techniques to enable the interactive visualization and exploration of large spatio-temporal data sets. The developed techniques include data management, processing, and rendering, facilitating real-time processing and visualization of large geo-temporal data sets. It includes three main contributions:

Concept and Implementation of a GPU-Based Visualization Pipeline. The developed visualization methods are based on the concept of a GPU-based visualization pipeline, in which all steps – processing, mapping, and rendering – are implemented on the GPU. With this concept, spatio-temporal data is represented directly in GPU memory, using shader programs to process and filter the data, apply mappings to visual properties, and finally generate the geometric representations for a visualization during the rendering process. Data processing, filtering, and mapping are thereby executed in real-time, enabling dynamic control over the mapping and a visualization process which can be controlled interactively by a user.

Attributed 3D Trajectory Visualization. A visualization method has been developed for the interactive exploration of large numbers of 3D movement trajectories. The trajectories are visualized in a virtual geographic environment, supporting basic geometries such as lines, ribbons, spheres, or tubes. Interactive mapping can be applied to visualize the values of per-node or per-trajectory attributes, supporting shape, height, size, color, texturing, and animation as visual properties. Using the dynamic mapping system, several kind of visualization methods have been implemented, such as focus+context visualization of trajectories using interactive density maps, and space-time cube visualization to focus on the temporal aspects of individual movements.

Geographic Network Visualization. A method for the interactive exploration of georeferenced networks has been developed, which enables the visualization of large numbers of nodes and edges in a geographic context. Several geographic environments are supported, such as a 3D globe, as well as 2D maps using different map projections, to enable the analysis of networks in different contexts and scales. Interactive filtering, mapping, and selection can be applied to analyze these geographic networks, and visualization methods for specific types of networks, such as coupled 3D networks or temporal networks have been implemented.

As a demonstration of the developed visualization concepts, interactive visualization tools for two distinct use cases have been developed. The first contains the visualization of attributed 3D movement trajectories of airplanes around an airport. It allows users to explore and analyze the trajectories of approaching and departing aircrafts, which have been recorded over the period of a month. By applying the interactive visualization methods for trajectory visualization and interactive density maps, analysts can derive insight from the data, such as common flight paths, regular and irregular patterns, or uncommon incidents such as missed approaches on the airport.

The second use case involves the visualization of climate networks, which are geographic networks in the climate research domain. They represent the dynamics of the climate system using a network structure that expresses statistical interrelationships between different regions. The interactive tool allows climate analysts to explore these large networks, analyzing the network's structure and relating it to the geographic background. Interactive filtering and selection enables them to find patterns in the climate data and identify e.g. clusters in the networks or flow patterns.

Zusammenfassung

Räumlich-zeitliche Daten sind Daten, welche sowohl einen Raum- als auch einen Zeitbezug aufweisen. So können beispielsweise Zeitreihen von Geodaten, thematische Karten die sich über die Zeit verändern, oder Bewegungsaufzeichnungen von sich bewegenden Objekten als räumlich-zeitliche Daten aufgefasst werden.

In der heutigen automatisierten Welt gibt es eine wachsende Anzahl von Datenquellen, die beständig räumlich-zeitliche Daten generieren. Hierzu gehören beispielsweise Verkehrsüberwachungssysteme, die Bewegungsdaten von Menschen oder Fahrzeugen aufzeichnen, Fernerkundungssysteme, welche regelmäßig unsere Umgebung scannen und digitale Abbilder wie z.B. Stadt- und Landschaftsmodelle erzeugen, sowie Sensornetzwerke in unterschiedlichsten Anwendungsgebieten, wie z.B. der Logistik, der Verhaltensforschung von Tieren, oder der Klimaforschung.

Zur Analyse räumlich-zeitlicher Daten werden neben der automatischen Analyse mittels statistischer Methoden und Data-Mining auch explorative Methoden angewendet, welche auf der interaktiven Visualisierung der Daten beruhen. Diese Methode der Analyse basiert darauf, dass Anwender in Form interaktiver Visualisierung die Daten explorieren können, wodurch die menschliche Wahrnehmung sowie das Wissen der User genutzt werden, um Muster zu erkennen und dadurch einen Einblick in die Daten zu erlangen.

Diese Arbeit beschreibt ein Software-Framework für die Visualisierung räumlich-zeitlicher Daten, welches GPU-basierte Techniken beinhaltet, um eine interaktive Visualisierung und Exploration großer räumlich-zeitlicher Datensätze zu ermöglichen. Die entwickelten Techniken umfassen Datenhaltung, Prozessierung und Rendering und ermöglichen es, große Datenmengen in Echtzeit zu prozessieren und zu visualisieren. Die Hauptbeiträge der Arbeit umfassen:

Konzept und Implementierung einer GPU-zentrierten Visualisierungspipeline. Die beschriebenen Techniken basieren auf dem Konzept einer GPU-zentrierten Visualisierungspipeline, in welcher alle Stufen – Prozessierung, Mapping, Rendering – auf der GPU ausgeführt werden. Bei diesem Konzept werden die räumlich-zeitlichen Daten direkt im GPU-Speicher abgelegt. Während des Rendering-Prozesses werden dann mittels Shader-Programmen die Daten prozessiert, gefiltert, ein Mapping auf visuelle Attribute vorgenommen, und schließlich die Geometrien für die Visualisierung erzeugt. Datenprozessierung, Filtering und Mapping können daher in Echtzeit ausgeführt werden. Dies ermöglicht es Usern, die Mapping-Parameter sowie den gesamten Visualisierungsprozess interaktiv zu steuern und zu kontrollieren.

Interaktive Visualisierung attributierter 3D-Trajektorien. Es wurde eine Visualisierungsmethode für die interaktive Exploration einer großen Anzahl von 3D Bewegungstrajektorien entwickelt. Die Trajektorien werden dabei innerhalb einer virtuellen geographischen Umgebung in Form von einfachen Geometrien, wie Linien, Bändern, Kugeln oder Röhren dargestellt. Durch interaktives Mapping können Attributwerte der Trajektorien oder einzelner Messpunkte auf visuelle Eigenschaften abgebildet werden. Hierzu stehen Form, Höhe, Größe, Farbe, Textur, sowie Animation zur Verfügung. Mithilfe

dieses dynamischen Mappings wurden außerdem verschiedene Visualisierungsmethoden implementiert, wie z.B. eine Focus+Context-Visualisierung von Trajektorien mithilfe von interaktiven Dichtekarten, sowie einer Space-Time-Cube-Visualisierung zur Darstellung des zeitlichen Ablaufs einzelner Bewegungen.

Interaktive Visualisierung geographischer Netzwerke. Es wurde eine Visualisierungsmethode zur interaktiven Exploration geo-referenzierter Netzwerke entwickelt, welche die Visualisierung von Netzwerken mit einer großen Anzahl von Knoten und Kanten ermöglicht. Um die Analyse von Netzwerken verschiedener Größen und in unterschiedlichen Kontexten zu ermöglichen, stehen mehrere virtuelle geographische Umgebungen zur Verfügung, wie bspw. ein virtueller 3D-Globus, als auch 2D-Karten mit unterschiedlichen geographischen Projektionen. Zur interaktiven Analyse dieser Netzwerke stehen interaktive Tools wie Filterung, Mapping und Selektion zur Verfügung. Des weiteren wurden Visualisierungsmethoden für verschiedene Arten von Netzwerken, wie z.B. 3D-Netzwerke und zeitlich veränderliche Netzwerke, implementiert.

Zur Demonstration des Konzeptes wurden interaktive Tools für zwei unterschiedliche Anwendungsfälle entwickelt. Das erste beinhaltet die Visualisierung attributierter 3D-Trajektorien, welche die Bewegungen von Flugzeugen um einen Flughafen beschreiben. Es ermöglicht Nutzern, die Trajektorien von ankommenden und startenden Flugzeugen über den Zeitraum eines Monats interaktiv zu explorieren und zu analysieren. Durch Verwendung der interaktiven Visualisierungsmethoden für 3D-Trajektorien und interaktiven Dichtekarten können Einblicke in die Daten gewonnen werden, wie beispielsweise häufig genutzte Flugkorridore, typische sowie untypische Bewegungsmuster, oder ungewöhnliche Vorkommnisse wie Fehlanflüge.

Der zweite Anwendungsfall beinhaltet die Visualisierung von Klimanetzwerken, welche geographischen Netzwerken in der Klimaforschung darstellen. Klimanetzwerke repräsentieren die Dynamiken im Klimasystem durch eine Netzwerkstruktur, die die statistische Beziehungen zwischen Orten beschreiben. Das entwickelte Tool ermöglicht es Analysten, diese großen Netzwerke interaktiv zu explorieren und dadurch die Struktur des Netzwerks zu analysieren und mit den geographischen Daten in Beziehung zu setzen. Interaktive Filterung und Selektion ermöglichen es, Muster in den Daten zu identifizieren, und so bspw. Cluster in der Netzwerkstruktur oder Strömungsmuster zu erkennen.

Acknowledgments

This thesis concludes my research, which I have been able to do as a research assistant in the Computer Graphics Systems Group at the Hasso Plattner Institute in Potsdam, Germany. I would like to thank Prof. Dr. Jürgen Döllner for giving me this opportunity. I would also like to thank Prof. Dr. Konrad Polthier (FU Berlin), Prof. Dr. Hartmut Asche (HPI, University of Potsdam), and Prof. Dr. Heidrun Schumann (University of Rostock) for agreeing to review this thesis.

Many thanks go to Dr. Thomas Nocke and his colleagues from the Potsdam Institute for Climate Impact Research (PIK) for a fruitful and long lasting collaboration on climate networks. I enjoyed working with you on this very interesting topic and thank you for your interest in my interactive application for your use case. I also wish to thank Deutsche Flugsicherung GmbH for providing the air-traffic data used in this work.

Further, I would like to thank Dr. Matthias Trapp for the collaboration on this research and many valuable discussions along the way. Also many thanks to the colleagues with whom I collaborated on several other interesting research projects. Especially, I would like to thank Daniel Limberger, Willy Scheibel, and Sebastian Hahn for their support and collaboration over the years.

Thank you also to everyone who has helped me by reviewing and giving me feedback on any parts of this thesis, I really appreciate it. Finally, thank you to my family for indulging me on this journey and for keeping me sane.

This work has been partially funded by the German Federal Ministry of Education and Research (BMBF), Germany, within the Potsdam Research Cluster for Georisk Analysis, Environmental Change, and Sustainability (PROGRESS).

Potsdam, Germany, November 15, 2018 *Stefan Buschmann*

Contents

Chapter 1

Introduction

The term *spatio-temporal data* refers to data that contains both spatial and temporal components such as geometry or locations changing over time. It occurs in an increasing amount of data sources and databases, for example time series of geographic data, geo-referenced social network interactions, or when tracking movements, which are often captured or generated continuously and in real-time.

Based also on the advances of today's sensors – as in the fields of remote sensing, sensor networks, or real-time image processing – spatio-temporal data represents an important data category for IT applications and systems. Consequently, efficient and fast techniques for storing, processing, analyzing, and visualizing spatio-temporal data more and more become substantial and required features in the respective domains.

This section first examines types and characteristics of spatio-temporal data. It then introduces selected concepts and examples of spatio-temporal visualization as well as real-time rendering methods, which are required for implementing interactive visualization systems. Geometric instancing and procedural geometry approaches are briefly described as an example of techniques that improve the performance of such interactive visualization systems. Finally, the problem statement and contributions for this thesis are presented.

1.1 Spatio-Temporal Data Characteristics

In the following, we introduce and characterize different categories of spatio-temporal data that are subject of this work. Based on that, requirements and challenges for the developed visualization framework are derived.

Multidimensional Data *Multidimensional data* refers to data that comprises multiple dimensions and represents, for example, data aggregates. In addition, it "can be described as arrays over heterogeneous data types together with meta data to describe them" [70]. Conceptually, each data item in such data aggregate can also be interpreted as a point in the corresponding n-dimensional data space. Based on that concept, individual data items can be interpreted as attributes of an object described and represented by a multidimensional vector. In statistics, multidimensional data is used to represent and store objects described by multiple — dependent or independent — variables. In this context, the data is also called *multivariate data*. In most applications managing big data, multivariate data plays a key role for the purpose of visualization and analytics.

Geospatial Data *Spatial data* denotes data containing spatial references that define positions and extents, such as points, lines, areas, or other geometries, for a given spatial reference system. It usually describes objects or phenomena of our physical environment. If the spatial reference system is the Earth, spatial data is also called *geospatial data*. It can be defined as "data and information having an implicit or explicit association with a location relative to the Earth" [89]. This spatial component is usually represented by a two or three dimensional vector describing the position of the data using earth-centric reference systems. For geographic data on a global scale, positions are often specified by latitude and longitude, and optionally the elevation (height), if 3D information is concerned. Other coordinate systems, such as local geographic coordinate systems that are defined for a specific area, are also common, especially for local and regional geo-data. In practice, geospatial data often also contains non-spatial components, which represent application specific data attributes (e.g., temperature, velocity, pressure). In a technical sense, geo-data is frequently encoded by multidimensional data.

Spatio-Temporal Data Spatial data often contains a temporal component and, hence, is called *spatio-temporal data* or *geo-temporal data*. Spatio-temporal data is used to describe objects or phenomena that change over the course of time. It can be defined as "multidimensional data, like points, line segments, regions, polygons, volumes, or other kinds of geometric entities that vary in the course of time" [191]. Many different categories of spatio-temporal data can be found in geosciences and geospatial applications. Time series, for example, usually contain multiple snapshots of the same data set, taken at different points in time; the temporal component is implicitly given. Another category is tracking-data, which describes the state of entities, often physical objects that are changing over time, for example sensor data or movement tracking data. A more general category are space-time events, which occur over time, but do not directly reflect changing entities. For example, social media events would fall into this category as messages occur over time and can be attributed with a position (e.g., the point of origin from the poster), but do not directly reflect physical objects that are changing or moving in physical space.

Movement Data While spatio-temporal data in general includes all data that contains both spatial and temporal components, *movement data* specifically describes movements of objects in a physical or virtual space. It is usually obtained via motion tracking, for example by tracking the movements of humans, vehicles, or animals. These instances of movement data can be categorized as *moving points* as they describe a position that changes over time. In other cases, more complex forms of geometry must be considered, such as *moving lines* or *moving regions*. This is the case for example to describe the dynamics of weather events, the movement of a flock of birds or other animals, or to monitor regions affected by a disease. Finally, the term *moving objects* considers not only a geometry that changes over time, but includes also the additional characteristics of a moving entity, expressed by complex attribute values that are also changing over time.

Trajectories A movement is represented by its *trajectory*, which is "the path made by the moving entity through the space where it moves. The path is never made instantly but requires a certain amount of time" [18]. Thus, a trajectory refers to the entirety

of a movement, which can be expressed by a sequential list of positions ordered by time. Trajectories are also often attributed, containing additional changing attributes, such as velocity, direction, or sensory data associated with the moving object. To efficiently store, retrieve, and query information about moving objects, *moving-object databases* [59] have been developed. They use spatio-temporal data models [143, 32] designed specifically for representing time-dependent data types, such as moving points, regions, and objects, and use spatio-temporal indexing methods and data structures [135] to efficiently store and access spatio-temporal data. Moving-object databases allow users to specify queries regarding both the spatial and the temporal domains, for example, to access the continuous positions of a movement, to determine the order of events, or in general to identify movement patterns [68].

Geographic Networks Networks or graphs are data structures in which objects are represented by nodes and relations between these objects are represented by edges. *Geographic networks* are geo-referenced graphs, in which node positions correspond to geographic locations. They can either express actual geographic features (e.g., transport networks such as roads or railroads, power grids, etc.) or non-spatial relationships between geographical positions (e.g., political or cultural relations between regions or information flows). Challenges for the analysis of such graphs lie in the combination of spatial and non-spatial information encoded in a geographic network. To enable the interpretation of data in its geographic context, the embedding of the network into a geographic map or geo-virtual environment is important. Due to the spatial positioning of nodes, however, the graph layout is fixed. Therefore, graph layouting techniques typically used in graph visualization to emphasize and express graph topology cannot be applied for geographic networks. This complicates perception and interpretation of network structures. Also, due to the geographic layout and embedding, problems of visual clutter and occlusion can occur, especially in the case of large networks.

1.2 Spatio-Temporal Data Analysis and Visualization

From a software engineering perspective, spatio-temporal data is often *complex* (i.e., it is multidimensional, as it contains a large number of attributes per data item) as well as *large* (i.e., it contains a large number of data items). This poses manifold challenges for processing and analysis. Methods for analyzing spatio-temporal data sets include statistical analysis [45], spatio-temporal data mining [153], and exploratory analysis combined with visualization methods [43]. They aim at revealing spatio-temporal patterns and thereby gaining insight into the spatial and temporal phenomena expressed by the data. In *confirmatory data analysis*, this is performed to confirm or disprove hypotheses, and to display the results for knowledge transfer, for example by statistical reports or visualization. *Exploratory data analysis* [16], on the other hand, enables users to interact with the data in order to detect and discover structures interactively. One use case are decision support systems: current events can be compared to the structures and patterns revealed in historic data, thereby being a basis for the current decision making

process, and to a certain degree allowing predictions into the future. In exploratory data analysis, visualization methods play a fundamental role to convey information to the users and allow them to examine the data.

The goal of visualization is to produce perceptible representations such as images or animations that help humans to extract information, to learn and understand facts, and to draw conclusions from data. Visualization can be used as a tool for the analysis of data as well as for the communication of analytical results. Visualization employs the human visual system, which by nature is very powerful with regard to recognizing patterns, detecting anomalies or outliers, and identifying clusters. Visualization techniques, for that reason, try to adapt themselves to the human system by presenting data in a way that helps the human mind to comprehend and understand aspects of the displayed data [195]. Spatio-temporal data visualization falls into the category of *scientific visualization* [71]. It is concerned with analyzing and communicating phenomena of the real world, in particular scientific data, which is often 3-dimensional and tends to be large in size.

Visualization in general can be defined as the process of deriving visual (more general: perceptible) representations from data. To describe this process, the *visualization pipeline* [71] concept has been introduced. It identifies three major steps: filtering, mapping, and rendering. First, the data is analyzed and prepared, and a subset of the data is selected for rendering (filtering). This often includes a preprocessing step that applies for example data validation, aggregation, or transformations on the data, and can also be interpreted as an additional step in the visualization pipeline (preprocessing). Then, geometric representations are derived from the selected data (mapping). Finally, these geometric representations are transformed into the final image (rendering).

To analyze large spatio-temporal data sets, exploratory analysis approaches are increasingly applied. As a result, *visual analytics* [44, 97, 9] emerged as a research field. It focuses on "analytical reasoning facilitated by interactive visual interfaces" [44]. Visual analytics integrates visualization with interaction and exploration-based methods, as well as data mining techniques that amend the data visualization with computational analysis results, enabling users to interactively explore and reason with the data.

Visual analytics is especially useful when dealing with large-scale, multidimensional data sets. By allowing users to explore data sets and interact with the applied visualization and analysis methods, the human perception and its extraordinary pattern recognition abilities are effectively employed to facilitate the understanding of structures and patterns in spatio-temporal data.

The main dilemma for the visualization of spatio-temporal data lies in the depiction of temporal information, which can be either explicit or implicit. To visualize temporal information explicitly, time must be encoded in a visual variable. However, only a limited number of visual variables are available in most visualizations, such as shape, position, size, color, and texture [31]. Time can be expressed for example by mapping it to a spatial axis, as it is perceived linearly and therefore can be easily understood as an analogy to time. This is applied in the concept of the *Space-Time Cube* [72], which depicts spatio-temporal data items by mapping time onto the up-axis.

As an alternative, time can also be depicted implicitly, using for example animation or interaction to display different states of the data sequentially. When the main focus lies on depicting changes in the data between two points in time, a *difference visualization* can also be appropriate. Such implicit methods have their limitations, as only one or few states are visible at any time, making it harder to perceive overall changes and trends. Also, to communicate larger developments over time, the results cannot be displayed by a single image but need to be displayed by, for example, a series of images or a video.

Another form of dealing with time is *aggregation*. In this context, an aggregation method accumulates data over time and generates a single result that represents the combined characteristics of the individual states. For example, bundling and spatial generalization techniques for trajectories [20] can reveal commonly used paths of all trajectories, and *density maps* can represent the amount of traffic at each position aggregated over a specified time period.

Movement trajectories can be visualized by a direct geometric mapping, using polylines to represent trajectories. They are placed within a geo-virtual environment, such as a 3D globe, a 2D map, or generally into the workspaces of a GIS [40, 60] to provide a reference system. To encode additional data attributes, such as direction, speed, or traffic volume, glyphs as well as colors are applied. The main challenges lie in visualizing large numbers of trajectories and the depiction of complex attributed trajectories. To visualize the temporal aspect of the movement, the *Space-Time Cube* concept is often applied, which maps time onto a visual axis. However, if true 3D movements are concerned, such as in the case of 3D aircraft movements, this cannot be applied as all three spatial axes are required to convey the spatial information.

Each of the described visualization methods is applicable for specific use cases and analysis tasks. Therefore, users must be able to combine different kinds of visualization, to enable them to effectively examine and analyze different aspects of the data.

1.3 Real-Time Rendering

Computer graphics refers to a vast discipline in computer science concerned with visual media and related processes and techniques for their generation and manipulation, for example for pictures, movies, games, or other interactive computer programs [62]. The process of generating and synthesizing visual media is called *rendering*, and it can be roughly subdivided into two categories: *non real-time rendering* and *real-time rendering* [3]. Non real-time rendering techniques, such as *ray tracing* [65], are generally used to achieve high-quality results in terms of physical light distribution and photorealistic appearance, but they are computationally expensive and, therefore, require significant resources in terms of memory and computation time. Real-time rendering techniques, on the other hand, are designed to generate images fast enough for interaction, but generally do not yet achieve the same level of quality and visual realism. Rendering speed is measured in frames per seconds (FPS), i.e., the number of images (frames) that can be rendered in a second. For interactive systems, rendering speed must reach a level

at which users can interact with the virtual environment and immediately see the result of their interaction. These are called *interactive frame rates*.

With today's computer hardware, real-time rendering is supported by dedicated graphics processing units (GPU). Those are massive parallel processing units following a SIMD (single-instruction-multiple-data) architecture, which can significantly improve rendering speed. They provide a rendering pipeline that is based on a rasterization approach: a scene, consisting of 3D geometric objects, are transformed for a given virtual camera, projected into screen-space coordinates, and finally rasterized onto the screen to compose a final 2D image. Nowadays, this real-time rendering pipeline is in large parts programmable [33] by shader programs, which can be applied to control each step of the pipeline programmatically. The individual steps of the real-time rendering pipeline can be described by three stages:

Application Stage The application stage describes the application logic that is responsible for preparing and controlling the rendering process. This can involve for example loading geometry, creating virtual scenes, generating and uploading data onto the GPU, and finally invoking the geometry processing and rasterization pipeline.

Geometry Stage In the geometry stage, each primitive and vertex is processed. A *vertex shader* is applied to modify each vertex, for example by applying transformations and projections. Optionally, a *geometry shader* can be applied, which can emit new geometric primitives for each vertex. This enables us to transform geometry or even generate new geometry on the fly. As the last step, geometry can be refined using *tessellation shaders*, that allow for subdividing patches in order to create geometry at higher detail. This step is also optional.

Rasterization Stage After the geometry stage, which defines all vertex positions in screen coordinates as its output, the GPU starts to rasterize the primitives onto the screen. This step can be controlled by a *fragment shader*, which determines the color for each fragment. In a typical geometric rendering process, this involves for example texture mapping, lighting, and shading.

In addition to the described real-time rendering pipeline, modern GPUs support an additional type of shader, the *compute shader*. It can be invoked outside of the rendering pipeline to perform general-purpose computations on the GPU (*GPGPU*) [142], employing its massive parallel architecture. Compute shaders can be used, for example, to transform or compute data, whose results are then again used in the rendering pipeline.

Visualization systems for spatio-temporal data must be able to process large numbers of data items. To perform this within real-time rendering constraints, methods for simplifying and reducing the size of data and geometry as well as optimized rendering techniques are required. *Level-of-detail* techniques [127] are an example for this. Direct visualization approaches for large data sets often represent data by using relatively simple geometries, such as rectangles, cubes, circles, or spheres. The resulting computer graphics scene typically contains many occurrences of the same geometry at different locations,

whose appearances can vary, but which can be defined as variations of the same basic geometry, applying for example different transformations, colors, or textures. For such cases, methods for sharing geometry between the rendering of their individual instances can improve both memory consumption and rendering performance.

Geometry instancing is a technique for drawing several instances of the same geometry at once [38]. This can drastically reduce the number of draw calls, which is an important factor contributing to the overall rendering time. Using instancing, the same mesh or geometry is rendered each time, but additional options can be passed on to the shader programs that can modify the appearance of each instance. For example, geometric transformations, textures, or colors can be modified this way.

Another approach lies in generating geometry procedurally instead of using predefined meshes. *Procedural geometry* [173] creates geometry algorithmically, using an implicit description of the geometry, for example defined by mathematical equations. By changing the parameters for the geometry-generation function, different instances of the same basic geometry can be generated.

1.4 Problem Statement and Contributions

Visual analytics for spatio-temporal data enables users to explore, analyze, and understand complex and large data. The main challenges are to handle the data sets, to provide visualization and analysis techniques that facilitate getting insights on the data, and to integrate suitable exploration and interaction techniques.

With regard to 3D geographic data, this becomes even more challenging. If three-dimensional information is important to the analysis of the data, 3D visualization techniques need to be applied. This often also requires to integrate visualization representations into 3D virtual environments to enable the interpretation of the data with respect to its location and surroundings. Therefore, specialized 3D visualization and interaction techniques are required.

The focus of this thesis is to design and develop components for visual analytics of 3D geo-temporal data and to provide a framework, such that these components can be used to assemble visual analytics systems, applications, and services (Figure 1.1). The thesis presents a concept for a GPU-based visualization pipeline that supports processing and rendering of complex geo-temporal data sets on the GPU and enables interactive visualizations. Based on that, visualization approaches for a number of specific use cases have been implemented to show the applicability of the concept. The framework is demonstrated by two use cases: visualizing 3D trajectories of aircraft movements and geo-referenced climate networks.

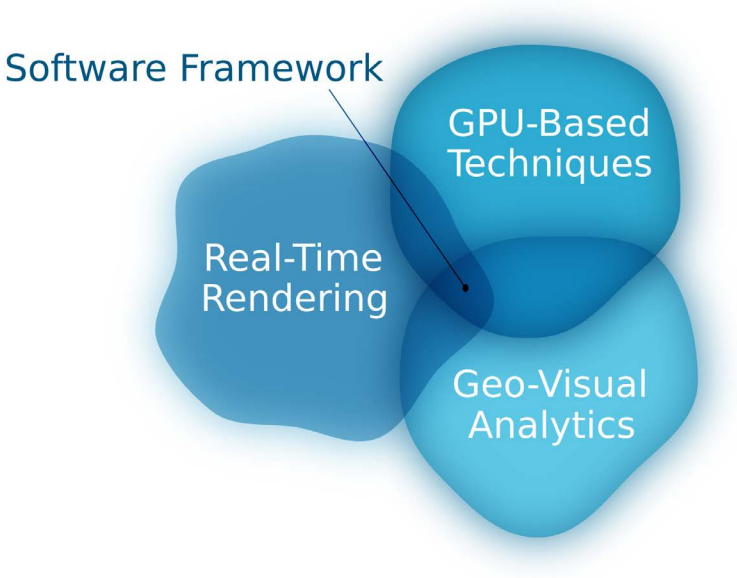


Figure 1.1: *This thesis touches on three main research areas: GPU-based visualization techniques, real-time rendering, and geo-visual analytics.*

Contributions

The main contributions of this thesis include:

Concept and Implementation of a GPU-Based Visualization Pipeline Visualization approaches for complex spatio-temporal data sets involve the depiction of large amounts of data items by geometric representations. In the visual analytics context, however, a high degree of interaction is required. This concept proposes an implementation that executes all stages of the visualization pipeline, i.e., filtering, mapping, and rendering, on the GPU. It applies a procedural geometry generation process that creates geometry in real-time rather than generating geometric representations on the CPU and uploading them onto the GPU for rendering. This allows for filtering and mapping operations to be applied interactively and improves performance by avoiding repeated uploads of geometry to the GPU.

Visual Analytics Framework for 3D Spatio-Temporal Data Based on this GPU-based visualization pipeline, various visualization and processing components for spatio-temporal data have been developed. They are designed to be used as building blocks for creating specialized visual analytics tools and have been integrated into a visual analytics framework. This includes a geometry-based visualization for complex geo-referenced data items, a visualization for large geo-referenced networks, an integration of geographic maps supporting dynamic geographic projections, and a real-time generation of density maps for aggregating data.

Visualization of Large Numbers of Attributed Movement Trajectories The framework has been used to implement a visualization technique for complex, attributed 3D trajectories that are integrated into a 3D geo-virtual environment. It uses direct geometric representations that can depict attribute values in shape, size, color, and transparency. In addition, texturing and animation are applied to communicate properties such as speed or direction. To explore and analyze the data, interactive filtering, mapping, highlighting, and temporal exploration methods are supported. Visual aids have been included to enhance the visual perception in 3D environments, including shadows and fences.

Visualization of Large Geo-Referenced Climate Networks The framework has also been used to implement a technique that visualizes 2D and 3D geo-referenced networks in a geo-virtual environment, supporting large numbers of nodes and edges. It supports interactive filtering of nodes and edges by their attribute values and dynamic mapping to visualize node and edge attributes. To facilitate the analysis and interpretation of global networks, the geo-virtual environment can be switched interactively between a 3D globe representation and several 2D map projections.

1.5 Thesis Structure

The remainder of this thesis is structured as follows. Chapter 2 gives an overview of previous and related work in the field of geo-temporal analytics. Chapter 3 presents the concept of the developed visual analytics framework for spatio-temporal data, based on the principle of a *GPU-based visualization pipeline*. A prototypical implementation of this framework based on OpenGL is presented in Chapter 4. This framework constitutes the basis for use case specific application examples presented in the following chapters. Chapter 5 describes the use case of air traffic analytics, particularly for analyzing large numbers of 3D aircraft trajectories. Chapter 6 presents climate network analysis as another use case, which is concerned with the interactive analysis of large geo-referenced networks. Chapter 7 contains a brief performance evaluation, discusses the accomplished results, and presents future research directions.

Chapter 2

Related Work

The following chapter presents related work in this thesis, which covers a number of different fields in computer science, geoinformatics, and data analysis. In the following, we outline most relevant works related to spatio-temporal data visualization, movement visualization, space-time cube, density maps and volumes, temporal exploration, and GPU-based visualization. In particular, this chapter summarizes visualization strategies for movement data and spatio-temporal data in general. A focus lies on the depiction of movement trajectories and the visualization of multivariate data, e.g., attributed movement trajectories or networks.

2.1 Spatio-Temporal Data Visualization

The research field of spatio-temporal data visualization examines methods to convey spatial and temporal information, both in static images and interactive systems. To understand and interpret spatio-temporal data, both the spatial and temporal aspects of the data have to be visually communicated and correlated to each other.

As a general technique for visualizing multidimensional data, *parallel coordinate plots* [91] can be used. They represent each axis directly as a column in the plot; the columns, which do not have an intrinsic order, can be arranged arbitrarily. A data point is represented by a line that crosses each axis at the corresponding data value. While this technique is able to convey multidimensional data values directly and accurately, it requires practice to interpret and suffers from perceptional problems, especially with large data sets, i.e., parallel coordinate plots are limited with respect to scalability [63]. Since a new column is needed for each data dimension, parallel coordinate plots do not scale well with a large number of data dimensions. They also do not scale well with large numbers of data items, as the visual clutter caused by overlapping lines makes perception and interpretation of plots difficult with growing data sets. Also, the temporal dimension is not treated differently to any another data dimension in a parallel coordinate plot.

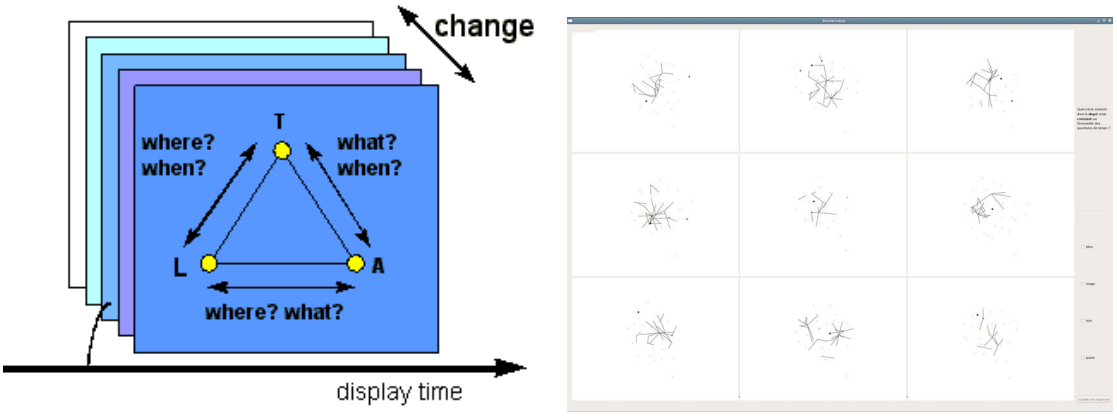
To reflect the unique role of time, particular spatio-temporal visualization methods have been developed. The major difficulty for such methods lies in the simultaneous depiction of spatial and temporal aspects. For this, different fundamental principles have been explored: animation, small multiples, linked views, and time geography [23, 16].

Using *animation*, the dynamics in a spatio-temporal data set are presented to users by displaying a sequential series of images [50]. In addition to the visual variables used in the individual images, animation adds dynamic variables, which can convey additional information, e.g., the duration and order in which the individual images are displayed [103, 129]. While animation enables the perception of changes and trends in a natural way, a direct comparison of the individual images and, therefore, a detailed analysis of spatio-temporal data is hindered. Therefore, the efficiency of animation for conveying complex information has been questioned [189, 152]. Ware et al. however showed that animation is helpful in visualizing flow fields [197].

Small multiples also use individual images for each point in time, but instead of animation, the resulting images are displayed alongside each other and are arranged in a chronological order. This enables a direct comparison of the specific states at different points in time and has been found to increase the understanding of spatio-temporal data as compared to animation [26]. However, as screen space is limited, only a small number of images can be displayed simultaneously. A comparison of both concepts, animation and small multiples, is shown in Figure 2.1.

Different aspects of the same data are often displayed using multiple *linked views* [151]. For example, one view may visualize geographic locations in a map, while a different view displays the temporal distribution, and a third shows multivariate data in a parallel coordinate plot. When users interact with one view, for example by means of filtering or selection, the result is immediately updated in all other linked views. This method enables the visualization of the different aspects of the data, while allowing users to correlate the information displayed by the different views.

Another approach for displaying space and time simultaneously is *time geography*. With this method, positional data is displayed on two spatial axes, while the third reflects time. The most common form is the space-time cube, explained later in this section. Figure 2.2 shows a visualization of a movement trajectory with the space-time cube.



(a) *Concept of displaying time-variant data by animation, showing individual images sequentially [103].*

(b) *Example of displaying time-variant data using small multiples that are arranged in a grid [26].*

Figure 2.1: *Time-varying data visualization using animation (a) and small multiples (b).*

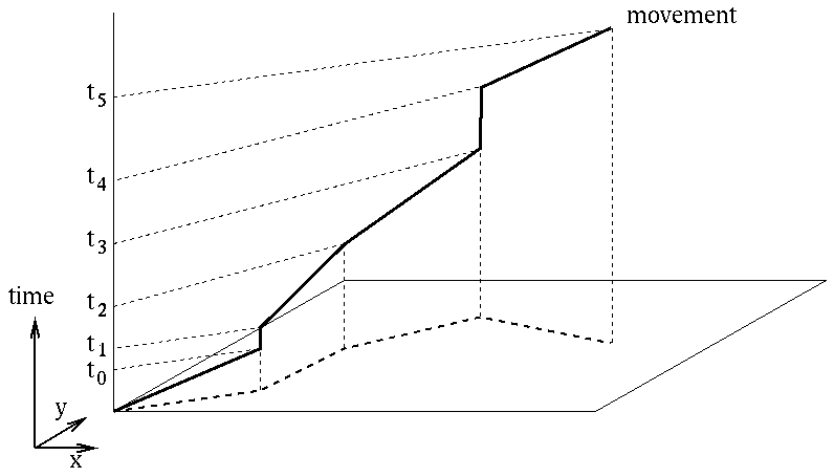


Figure 2.2: *Visualization of a movement trajectory using the a space-time cube [186].*

As an orthogonal categorization, visualization methods can also be divided into direct depictions and aggregated views [10]. Direct visualization approaches represent data elements by individual geometric objects. For example, movement trajectories can be depicted by lines or more complex geometries such as ribbons or tubes, and individual data items may be represented by geometries or glyphs, providing visual variables such as shape, size, or color to convey multivariate data.

In contrast to displaying each data item separately, aggregation methods derive a generalization from the data that represents a summary of multiple or all data items. Examples of this are spatial clustering algorithms [10], as well as density based methods [202], which aggregate the influence of spatio-temporal events in a spatial area. For example, to analyze large numbers of individual movements, flows can be extracted, which represent the most common movement patterns. Such visual summaries can display overall trends and clusters in the data. They also enable a comparison of individual data items to the summary in order to find uncommon patterns and outliers.

2.2 Movement Visualization

The analysis of movement data represents an important task in many applications, such as traffic management and control, disaster management, crime investigation, or animal tracking. Hence, a number of different approaches for the visualization of movements and movement trajectories exist, which can display the path of moving objects and additionally depict their complex attributes.

Movement trajectories are often depicted using a direct visualization approach, in which trajectories are represented by lines (e.g., [5, 87]). To visualize multivariate data, more complex geometries such as tubes or ribbons have been used. Trajectory visualization is usually embedded into a geospatial map and combined with interaction techniques such as selection, filtering, and brushing to support interactive data exploration.

Krüger et al. proposed an interaction technique called TrajectoryLens, which supports a dynamic filtering of trajectories and enables users to formulate complex spatio-temporal queries with spatial lenses [113].

Ware et al. have demonstrated a multivariate visualization of trajectories displaying the underwater movement of humpback whales [196]. They use a ribbon geometry to convey several attributes (Figure 2.3): the center points encode the whale's position, direction is depicted by applying an arrow-shaped texture. The orientation is depicted by twisting the ribbon around its direction vector and an additional color-coding to highlight side roll behavior. Additional geometric patterns have been added to visualize the whale's fluke strokes.

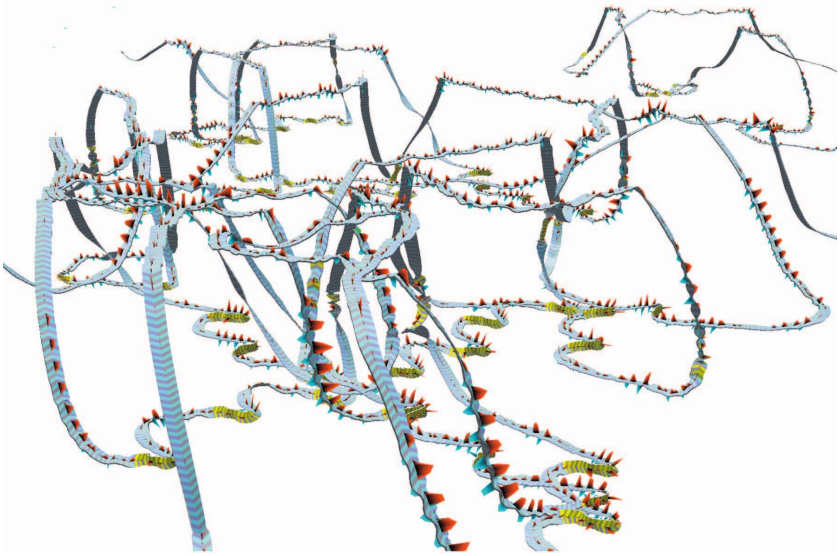


Figure 2.3: *Visualization of the underwater movement of whales using 3D ribbon geometries, texturing, and colored glyphs [196].*

Schirski et al. implemented a method for rendering 3D tubes to visualize particle trajectories. To improve rendering performance, they applied a billboarding technique to generate the visual appearance of three dimensional tubes, while using much simpler geometries [160]. Since rendering performance as well as dynamic geometry generation capabilities have greatly improved in current computer graphics hardware, such optimizations should not be necessary anymore. Instead, tube geometry can now be generated directly on the GPU using geometry and tessellation shaders, still achieving interactive frame rates even for large data sets.

To visualize attributes of multidimensional data, the concept of visual glyphs [129] can be used. Those are graphical representations placed on a map, which can vary in form, color, or size, to express multivariate data. Ward provided a taxonomy for the placement of glyphs in multivariate data visualization [193]. Interrante worked on the perception of texture patterns to derive a texture palette for the visualization of multivariate data [92].

Glyph-based visualizations have been applied to convey time dependent data in the context of geographical maps [181, 178]. For example, Bak et al. used glyphs, called growth ring maps, to convey both spatial and temporal information acquired by sensor logs of mice movements [29]. Nienhaus et al. used dynamic glyphs to depict dynamics following visual art and graphic design principles [140]. Tominski et al. have used a wall-like approach to visualize spatio-temporal data in a 2D geographical context [180]. Guo at al. used embedded glyphs within the ThemeRiver metaphor [73] to visualize traffic data at road intersections, combined in linked views with direct trajectory visualization and parallel coordinate plots [69].

In the context of movement trajectories, the visualization of attributes is an important functionality. To visualize attribute values together with spatial and temporal information, Tominski et. al. used a stacking-based approach [183, 6] (Figure 2.4). Andrienko et al. have applied the approach to the space-time cube to visualize attributes of trajectories following similar routes [14].

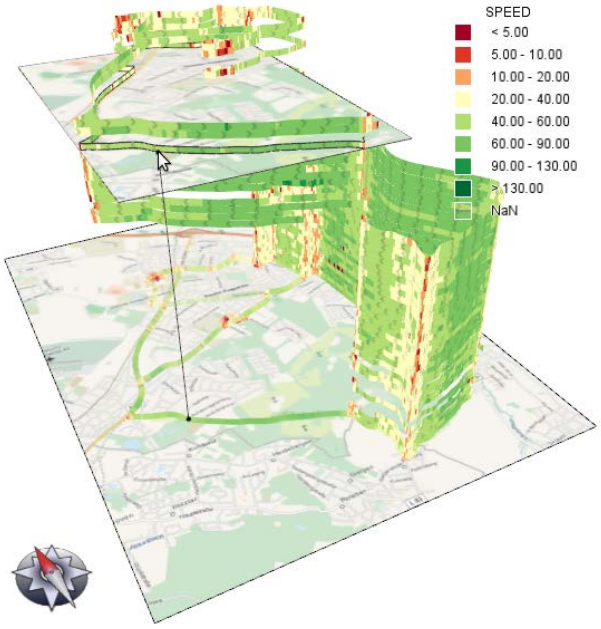


Figure 2.4: *Stacking-based visualization of trajectories [183].*

Wang et al. visualized sparse trajectory data, in which vehicles were recorded only when they passed through certain transportation cells and therefore the exact movements between the cells were not included [192]. They depicted cells as circular nodes and used color, size, and arrows to display traffic volume, speed, and flow direction. In addition, linked views and animation was applied to visualize the traffic flow. Bast et al. visualized large transit networks in real-time by means of glyphs that display a large number of vehicles at one point in time, and used animation to convey the changes over time [30]. Lange et al. used 3D tubes, glyphs, and animation to display movement trajectories and applied artist-designed encoding for attribute mapping [120].

Further techniques for the visualization of movement data are not explained in detail here, since their scope and use cases differ significantly from the use cases focused on in this thesis. For example, *flow trees* [206] and *OD maps* [205] are concerned with the origin and destination of movements, rather than the actual movement trajectories. Similarly, movement visualizations such as proximity-based trajectory visualization [46] regard derived values like distance to a given destination. Parameter-dependent movement analysis aims at examining the processes behind the movements, by extracting high-level features of movements and relating them to their respective parameter configurations [126]. Other techniques related to movement visualization are for example *flow visualization* [200, 177], or *treemaps* [172].

The visualization methods in this thesis use approaches for direct trajectory visualization that are similar to those presented above. Trajectories are displayed as line segments or more complex geometries such as 3D ribbons or tubes, embedded into a virtual 3D geographic environment. Due to the real-time generation of geometry on the GPU, geometric representations can be configured or exchanged rapidly. An interactive mapping process allows for trajectory attributes to be mapped onto visual variables, while texturing and animation are supported as additional means to convey information. Additionally, the methods are integrated into a data-driven and highly interactive pipeline that enables selection and combination of different visualization methods based on data input, analysis results, and interaction.

2.3 Space-Time Cube

The *space-time cube (STC)* is a visualization method proposed by Hägerstrand [72] that uses a three-dimensional representation of space and time: the x- and y-axes are used for the spatial components of the visualized data, while time is mapped to the z-axis (up-axis). This approach allows for creating an intuitive visualization of 2D spatio-temporal events using 3D visualization, since both the spatial and temporal aspects are depicted by spatial axes, thus, no iterative images or animations are necessary to display temporal data. As an example, Figure 2.5 shows a space-time cube visualization of "Napoleon's March on Moscow" (Minard's Map) [100].

In recent years, the space-time cube has become a popular visualization method in the field of geovisualization and geo-visual analytics, and several frameworks based on the STC have been developed. Kraak applied the space-time cube to modern computer-based visualization systems [101, 100, 105] and explored the implications that real-time visualization and interaction have on the concept. As a result, interaction methods for the space-time cube and temporal exploration techniques [102] have been developed. Kapler and Wright used a similar approach (*3D Timelines*), in which a spatial map is used as a reference point and spatio-temporal events are placed above and beyond the map to indicate past and future events [94].

Andrienko et al. compared the STC to other exploratory techniques, such as map iteration, animation, and querying [16]. They demonstrated the use of the STC for visualizing spatio-temporal events, such as data about earthquakes, in their interactive

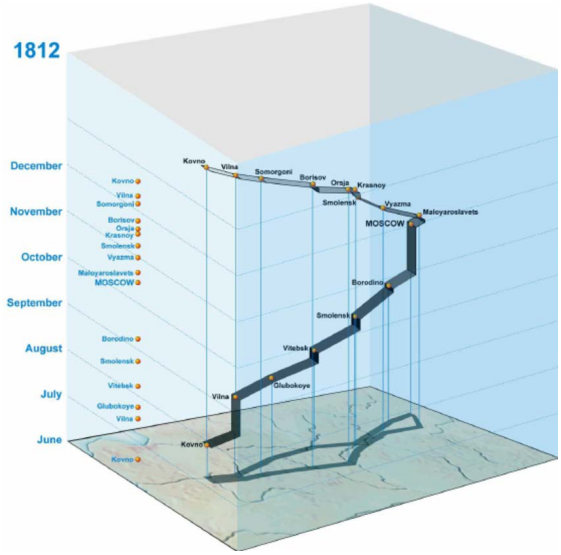


Figure 2.5: *Visualization of Minard's Map in a space-time cube [100].*

system CommonGIS [17] and examined temporal exploration techniques such as temporal focusing required for an interactive exploration of large data sets [64].

The space-time cube has been used particularly for the interactive visualization of movement data. Movement trajectories are thereby represented as space-time paths in the STC, projecting their two-dimensional positions on the two spatial axes, while mapping time on the third axis. Andrienko et al. developed a framework for the analysis or large amounts of movement data [15] that combines interactive visualization with computational methods, such as clustering of trajectories, and interactive temporal filtering. They demonstrated the use of interactive space-time cube visualization for a number of analysis tasks for trajectory data, for example to analyze interactions between moving objects [24], or to identify events by detecting significant changes in trajectories [12]. They also developed methods to facilitate the simultaneous analysis of multiple trajectories by applying time transformations to align the start and end times of trajectories in order to make is easier for humans to visually compare trajectories [4], and developed methods to analyze group movements [21]. To visualize and compare data about traffic jams, they applied temporal transformation as well as attributed trajectory visualization by color mapping in the STC [14].

For the analysis of large numbers of trajectories, clustering methods for trajectories have been applied [150] and combined with STC visualization [19]. Clusters of trajectories that follow the same path are identified by combining classification methods and human interaction [13, 7]. The resulting generalized trajectories are used to display a visual summary of the clustered trajectories [20] (Figure 2.6). A similar approach in the context of GIS has been applied by Shaw et al., which creates generalized space-time paths by identifying the centers of clustered trajectories [164].

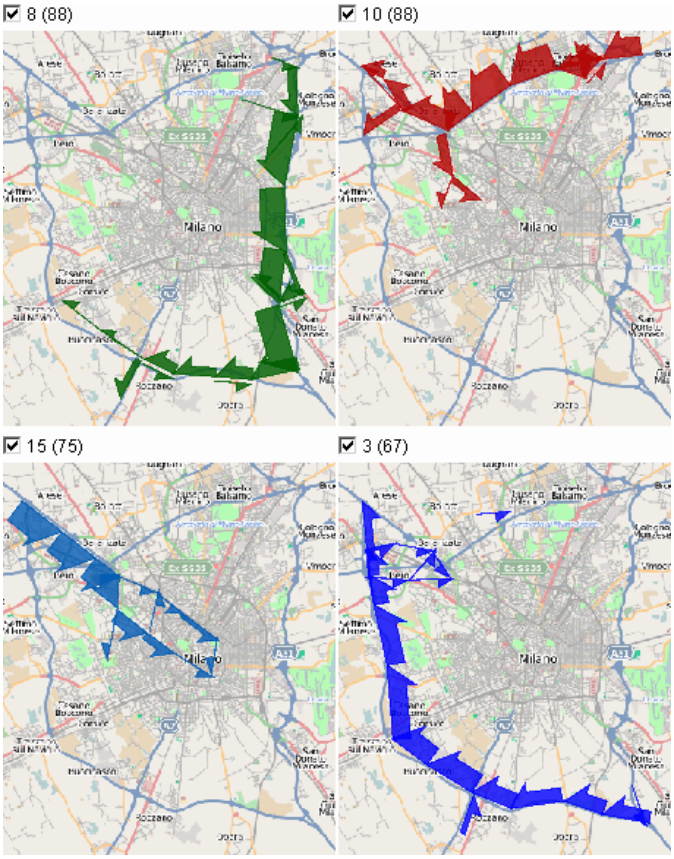


Figure 2.6: *Example of visual clusters of trajectories that follow the same routes. [19].*

Due to the popularity of the space-time cube especially for the visualization of movement data in interactive systems, the question arises how useful it actually is for communicating spatio-temporal data as compared with other visualization techniques. Kristensson et al. have conducted a user study on the space-time cube to provide an empirical basis for its use for spatio- temporal data analysis. They found that for complex analysis tasks, the use of the space-time cube resulted in approximately the same error rate but twice as fast response times as compared with a 2D representation of the same data [108, 107]. Kjellin et al. compared 3D visualization methods such as the space-time cube with 2D methods, concluding that structures that "remain invariant over affine transformations" can be perceived better in 3D visualizations, while otherwise 2D visualization leads to better results [98]. Kveladze et al. conducted a user study with domain experts focused around cartographic design with regard to the use of visual variables and depth cues in the space-time cube [115]. They developed a methodological framework for evaluating the space-time cube based on real-world data and systematic user studies "with special attention for design aspects and the environment in which the STC has to function" [116]. In another user study, they applied the space-time cube for the interactive exploration of movement data, comparing two user groups (domain

experts and non-experts). They found that the domain experts performed more effectively and efficiently in operating the provided visual analytics system [117]. Gonccalves et al. compared space-time cube and static map visualization for the analysis of human movement trajectories, concluding that the static map performs better for locating-tasks, while the space-time cube seems to better support association-tasks [66].

The space-time cube has been applied in many different application domains and use cases. Among them, movement data analysis [19] is one of the most prominent and has been applied in many fields, such as human, animal, or traffic movement analysis. As an example for a visualization of the effects of movements rather than direct movement analysis, air pollution in urban areas has been depicted using the space-time cube [61]. Another prominent use lies in the visualization of geo-referenced space-time events, such as social media events [104]. Other examples include the visualization of diseases in epidemiology [106], the storage and analysis of archaeological events [78], as well as eye-movement analysis in the context of user studies [121, 8, 147, 114]. Bach et al. have also used the space-time cube in a generalized way by depicting adjacency matrices for the visualization of dynamic networks [27].

By displaying spatial and temporal aspects simultaneously, the space-time cube makes it possible to determine and compare spatial and temporal overlaps, temporal order of events, or the length of events in a single visualization. Significant events, such as the intersections of tracked objects in time and space, can also be perceived relatively easily. On the other hand, space-time cube visualization suffers from the shortcomings of 3D visualization in general, such as occlusion of objects and perspective distortion, which have to be mitigated. The perception of depth for example can be improved by interactive camera control or by inserting additional visual depth cues, e.g., using shading techniques or artificial shadows [194, 115].

The space-time cube visualizes two-dimensional data by mapping the temporal dimension to the up-axis. It is therefore suitable especially for the visualization of two-dimensional data with a temporal component. To visualize true 3D data, however, the space-time cube cannot be directly applied: since the data itself is three-dimensional, all three axes are required for displaying the spatial components of the input data, therefore time cannot be mapped to one of the visual axes.

In this work, the space-time cube concept is applied as a mapping configuration that is available for direct trajectory visualization (see 5.8). For the use case of visualizing 3D movement trajectories, which requires the analysis Of movements in all three dimensions, the third dimension is used by default to visualize the altitude of moving objects. To support a detailed temporal analysis of selected trajectories and their temporal relationships, however, time can be interactively mapped to height. Similarly, height can also be used for the mapping of other trajectory attributes, to enable a multivariate visualization of trajectory attributes in a 3D view. Since attribute mapping can be applied and modified interactively, different mapping configurations can be defined for specific analysis tasks, for example, 3D trajectory visualization for a spatial overview and space-time cube for a detailed temporal analysis.

2.4 Density Maps and Volumes

Density maps can be applied to aggregate movements using a regular grid and to represent the influence of movement trajectories on their neighborhood in the form of a spatial map. Willems et al. presented an approach for using *kernel density estimation (KDE)* to aggregate vessel movements in a harbor [203, 202] (Figure 2.7). They proposed to apply different kernel sizes to distinguish between historical and current movements, which enables users to identify current movements and compare them with long-time movement patterns. This way, untypical and potentially dangerous movement behavior can be detected. Peters et al. used directed kernels to produce density maps that take into account the direction and speed of moving entities [144]. To visualize and explore quantitative differences between multiple density maps, Lampe et al. introduced interactive difference views calculated by subtracting KDE plots from each other [118].

Scheepens et al. introduced interactive density maps computed in real-time by utilizing modern graphics hardware and enabled the exploration of trajectory attributes by varying and combining multiple density fields. By selecting subsets of trajectories and adapting the parameters for the density calculation, users can explore different subsets of the data [159]. They extended their approach to composite density maps, which allow users to define the parameters of the density maps by combining and customizing building blocks [157]. The interactive density map approach has been applied and evaluated for several use cases, including vessel movement analysis and urban planning [158].

In combination with particle simulation to show dynamics, an extensive framework for analyzing traffic flows based on density maps has been developed [156]. Willems et al. conducted a user study to compare the density based approach for movement visualization to two other methods – animation of moving dots and the space-time cube. Their results showed that density maps performed better in identifying stopped objects while performing equally well in all other tasks [201].

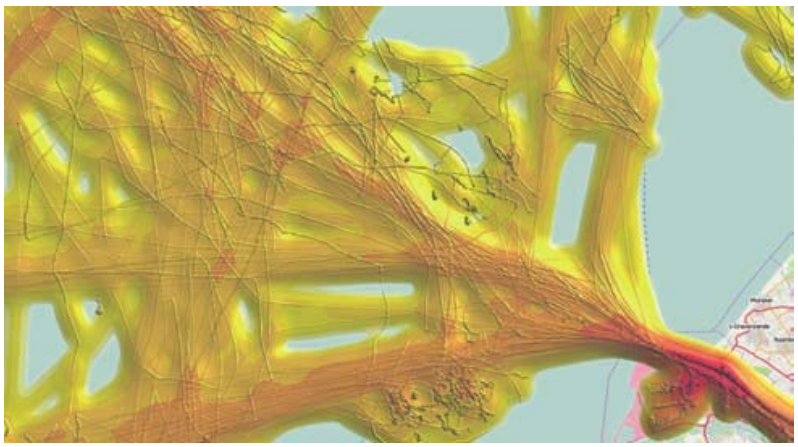


Figure 2.7: *Density of vessel movements on the dutch coast using kernel density estimation. [202].*

Similar to 2D density maps, the application of KDE for the aggregation of movement data can be extended to 3D. Demšar and Virrantaus have applied this method to trajectories in a space-time cube to counter the effect of visual clutter introduced by displaying large numbers of trajectories [49] (Figure 2.8). A similar approach has been applied for the mapping of crime scenes [139] and for the analysis of surveillance data, called *spatio-temporal trajectory volumes (STTV)* [93]. Eaglin et al. have applied this approach in a web-based visual analytics tool to analyze spatio-temporal data such as fever outbreaks and hurricane simulation data [55]. As a variation to general 3D space-time density, Demšar et al. developed a stacking based approach in which each horizontal layer is calculated separately and the results are stacked [48]. Wolff and Asche applied KDE for spatio-temporal analysis of crime scenes [204], using a 3D terrain metaphor to visualize crime hotspots in a German city and combining it with space-time paths to visualize the temporal chronology of individual incidents.

While those applications use KDE for space-time-paths in a space-time cube, the method can also be applied directly to movement trajectories that contain a spatial 3D component. For example, Bürger et al. used 3D density volumes to visualize uncertain particle trajectories in 3D flow fields, which they called *3D visitation maps* [36]. The resulting density volume represents the number of trajectories passing through each voxel.

In this work, density-based methods are applied in conjunction with direct trajectory visualization. 2D density maps as well as 3D density volumes are generated in real-time using the same input data as the geometry-based visualization methods. Therefore, KDE parameters, filtering, and mapping options can be adjusted dynamically and interactive temporal exploration can be applied. To compare different temporal selections, difference maps can also be calculated in real-time. The resulting density representations can be visualized separately or in combination with direct trajectory visualization. This enables advanced visualization metaphors such as *focus+context* or *overview and detail* methods.

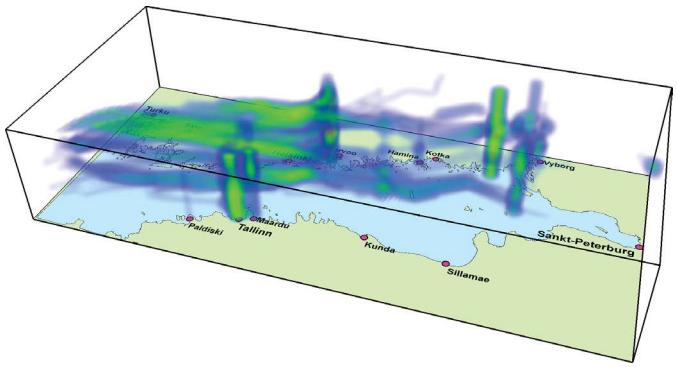


Figure 2.8: *Space-time density of tanker movements displayed by volume rendering techniques. [9].*

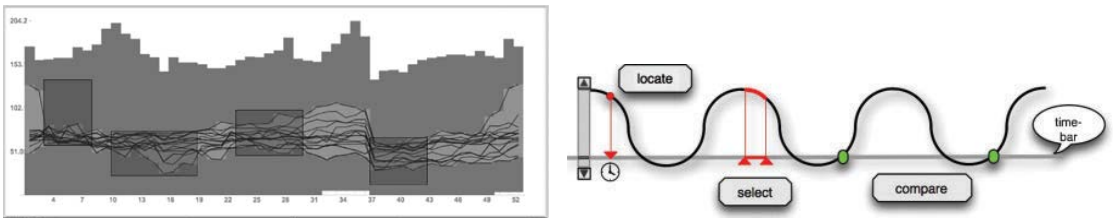
2.5 Temporal Exploration

Interactive visualization systems are enabled by visualization methods and real-time rendering techniques that produce images fast enough to run at interactive frame rates. Yet, to effectively explore and analyze data with such systems requires not only efficient data visualization methods, but also user interfaces and interaction techniques that allow users to express their queries into the data and thereby explore it interactively.

Shneiderman introduced *dynamic queries interfaces* as a way to formulate database queries with graphical user interfaces instead of using query languages like MySQL as a user interface – by adjusting graphical elements such as sliders or buttons, database queries are adjusted interactively and the resulting visualization is updated rapidly [165]. The same principle is still applied in interactive systems today.

Those interfaces can also be applied for temporal selection, interpreting time as linear integral value. A slider may be used to select a single point in time, for example to explore instantaneous events. To select time ranges, often a timeline is displayed, representing temporal selections by boxes, which can be manipulated to select start and end time. Hochheiser and Shneiderman proposed the *timebox* widget, which extends this principle to two dimensions [75]: the time period is selected on the horizontal axis, the vertical axis is used to filter by value.

Time, however, also has a circular or periodic characteristic, which cannot be expressed by pure linear selection tools. Edsall and Peuquet examined graphical user interfaces for the integration of time into GIS [56]. To select time linearly, they used an interactive scrollable *timeline*, which is applied independently for each layer, e.g., to explore different data attributes independently, but could also be linked together. As a second tool for temporal exploration, they introduced the *time wheel*, which enables the selection of a specific duration within the selected time cycle, for example a specific part of a year or time per day. A combination of the timeline and time wheel representations, the *time wave*, has been proposed by Li and Kraak [122]. It simultaneously displays the linear and circular aspects of time as a linear plot and can be applied for visualization or as an interaction tool. Figure 2.9 displays examples for the timebox and the time wave widgets as interaction tools for temporal data.



(a) *Timebox widget to select multiple ranges by data value and time [75].*

(b) *Visualization and interaction concept of the time wave to express temporal data [122].*

Figure 2.9: *Two interaction tools for selecting time intervals: timebox (a) and time wave (b).*

A common technique for the visualization of temporal data is the spiral view, in which data is displayed along a spiral, thereby depicting both the serial and periodic characteristics of time [37, 199]. The principle has also been extended to 3D [74], using a helix spiral as a 3D glyph that can be placed on a map. This method allows for periodic events to be identified visually, for example, events that occur around the same time each day. Tominski and Schumann applied the spiral also as an interaction technique [182]: to browse through time on the spiral, small buttons have been added to both ends of the spiral, as an alternative, the keyboard can be used to move in time (Figure 2.10).

Another way of visualizing and interacting with multi-dimensional data is by means of parallel coordinates plots [91, 90]. To highlight the central role of time in spatio-temporal data sets, Tominski et al. proposed the *TimeWheel*, in which axes are placed in a circular layout rather than parallel to each other, and the time axis is placed at the center [179]. This enables users to explore the relations between attribute values and time values when interacting with the data.

Andrienko et al. describe the *timeline filter*, an interactive visual tool that lets users define complex filters in a visual way (Figure 2.11). It displays time on the x-axis, while attribute values are displayed on the y-axis. By defining the boundaries of attribute values (e.g., selecting or deselecting categorical attribute values, or defining attribute ranges for numerical values), and specifying temporal intervals, complex filter conditions can be described [25].

In addition to interactively defining temporal as well as attribute filtering, a strong integration and combination of explorational tools with the applied visualizing techniques is required. With the principle of linking and brushing, different visualization views are connected and the results of interactive filtering and selection are updated in all linked views [96, 198]. Another challenge lies in the depiction of the selected data ranges in a visualization. Instead of filtering out unselected items, distinct visual appearances can be applied to represent selected and unselected data.

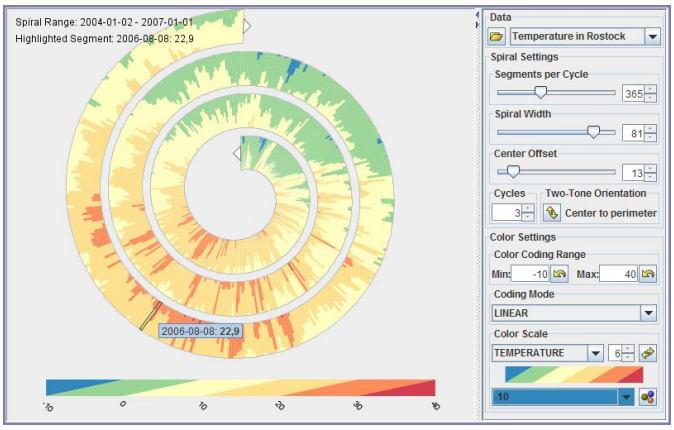


Figure 2.10: *Interactive spiral visualization enables users to detect cyclic patterns in data [182].*

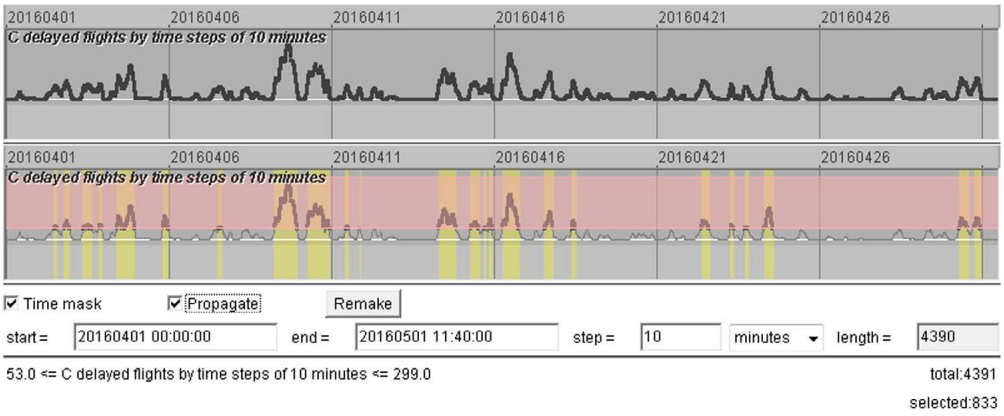


Figure 2.11: *Interactive timeline as a tool to select time ranges and attribute boundaries [25].*

Carvalho et al. proposed a temporal focus+context visualization model that defines a temporal degree of interest for the visualized items based on the selected temporal focus [39]. During mapping, it is used to alter the visual appearances of items to help distinguish between the temporal focus and context.

The framework developed in this thesis applies the principles of interactive filtering and exploration as described in this section. Data can be selected interactively by temporal as well as attribute-base filtering, using a linear representation of time. A temporal focus+context model has been applied that helps to distinguish between selected and unselected items, given one or several temporal foci. This information can be accessed in the interactive mapping step, which allows for different visual representations to be chosen based on the status of an item, for example to apply distinct visual representations for items in a focus area, an overlapping area, or unselected items. This allows for complex visual analysis functions to be defined and applied interactively.

2.6 GPU-Based Visualization Methods

Geo-visual analytics focuses on the integration of visualization techniques into interactive systems, by combining visualization, analytical methods, and interaction techniques to enable the development of exploratory analysis tools. Incidentally, GPU-based rendering and processing techniques have advanced significantly, and can be applied to enable the processing, visualization, and interaction with large data sets using the GPU.

In recent years, computer graphics hardware has advanced rapidly. Not only the performance has increased immensely, but also the flexibility and programmability of the graphics hardware has been improved. With the programmable graphics pipeline, most parts of the rendering pipeline are now programmable, making it possible not only to adapt the rendering process for specific purposes, but also to utilize the power of the GPU for general purpose computations [141].

These capabilities of modern GPUs can also be applied in the visualization processes. However, since the development of GPUs has been mainly motivated by gaming and simulation, rendering architectures have been optimized for those use cases. In today's graphics hardware, the rendering pipeline is optimized towards the efficient rendering of large geometries and scenes. This requires a predefined graphical scene, which is usually prepared on the CPU, optimized, and then transferred to be rendered efficiently by the GPU. Shaders are applied that refine the rendered geometry, perform animations or simulations, and apply visual effects.

Visualization, on the other hand, usually has a data-centered approach in which the visual scene does not necessarily exist prior to the visualization process. As part of the visualization, data is processed and filtered, then visual representations for the data are generated and configured, often influenced by user interaction. Finally, the resulting scene is rendered onto the screen. Although the programmable graphics pipeline is now capable of implementing this approach, it is seldom applied in this way.

Early works have identified the potentials of applying GPU-based techniques specifically for visualization and described concepts for defining visualization processes with the GPU [131, 28]). In many domains and use cases, GPU-based methods for data visualization have been applied, that improve rendering performance and in some areas even made interactive visualization possible.

In volume visualization [95], for example, large volumetric data sets need to be processed and rendered. Direct volume visualization techniques [47, 190, 176] use the texture acceleration capabilities of early computer graphics hardware to achieve interactive frame rates. Other sophisticated methods such as GPU-based ray casting [112, 174, 133] have been applied to improve both performance and quality. The GPU-based visualization architectures in these cases use 3D textures to represent data, and rasterization-based approaches for visualization. Per-pixel operations are used to implement transfer functions that map data values to their visual representations (e.g., color), and to apply graphical effects such as lighting and shading.

In terrain rendering [77, 41, 125, 161, 124], large 2.5D surfaces, often represented by raster-based height fields, are displayed. This requires a transformation of raster-based information into geometric representations. To generate smooth surfaces at interactive frame rates, adaptable grids and meshes have been developed, which are dynamically refined when view parameters have changed. Level-of-detail techniques have been applied that facilitate high-quality results, while reducing geometric complexity in order to maintain interactive frame rates. GPU-based techniques applied in such visualization methods include efficient storage and data management for raster-based data, i.e., height fields, as well as generative and adaptive geometries that are dynamically refined to correspond to the desired levels of detail.

Geovisualization involves the display of complex geo-data and its integration into 3D geo-virtual environments, such as virtual 3D city models or terrain geometry. This often requires large amounts of data and geometry for the rendering of both the data and its environment. Also, visual representations need to be flexible enough to enable techniques such as level-of-detail and level-of-abstraction, which allow users to display

(a) *Interactive rendering of 1.000.000 molecules represented by GPU-based compound glyphs [67].*

(b) *Visualization of atoms and interconnections using ellipsoids and stylized line primitives [175].*

(c) *Labeled tuboid rendering [145].*

Figure 2.12: *Examples for dynamically generated geometry on the GPU: compound glyphs (a), stylized lines (b), and tuboids (c).*

and explore data at different levels of complexity, based on interaction. To meet such requirements, GPU-based techniques for focus+context visualization of geo-data have been developed [185]. In specialized areas, such as edge bundling [76] for networks or graphs, image based methods have been developed on the GPU that enable real-time bundling of large graphs [82]. Also, immersive technologies are increasingly employed for the visualization and exploration of geographic data, for example for movement trajectories [85] or flow maps [208].

Also in other fields of data visualization, complex geometries need to be displayed. For example, to visualize molecules and proteins, highly complex geometric representations are required. Optimized rendering methods have been developed to visualize large sets of molecules at interactive frame rates [67]. To reduce the amount of memory required for data representation, generative geometry approaches have been applied, which represent molecules implicitly (e.g., storing only position and rotation data) and use geometry shaders to generate the geometries on the fly [119]. A similar approach has been used for visualizing proteins at a higher level of abstraction, called the secondary structure representation, which is used for illustrative purposes. In this abstraction, the protein structure is represented by illustrative figures such as ribbons and tubes, which can be generated on the GPU, thereby reducing the amount of memory needed to store large data sets [109]. GPU-based ray casting techniques have been applied to visualize surface representations of molecules and for time-dependent molecular simulations [110, 111].

When generating geometry on the GPU, often simple geometries, such as lines or splats, are used and methods such as texturing and shading are applied to make the visual impression of more complex geometries. Such methods have been applied for example for stylized lines [175] or tuboids [145], which helps to improve rendering performance and decrease memory consumption. Figure 2.12 shows an example of these techniques.

In recent years, works on using the GPU for information and scientific visualization have drastically increased. McDonnel and Elmqvist, for example, have proposed a refinement to the standard visualization pipeline model to better fit GPU-based implementation paradigms by adding a final image-based sampling step to the model [132].

Frameworks and systems for information visualization also take advantage of GPU-based implementations to optimize processing and rendering performance as well as to support the visualization of larger data sets. The Visualization Toolkit [162], for example, has been extended with GPU-based methods for scientific visualization purposes. The web-based visualization system imMens [123], which focuses on visual querying of big data, is based on shader programs for both data processing and rendering. Riccio and Lilley introduced the "programmable vertex pulling rendering pipeline", which applies programmable draw dispatches based on OpenGL 4 compute shaders to move more tasks in rendering complex scenes from the CPU to the GPU [149]. AVIST [134] proposed a GPU-centric design for data processing and visualization to support interactive filtering and visual exploration of large data sets. Stardust [148] is a web-based framework that aims at utilizing GPU-based implementations for efficient rendering, while providing a familiar API for the development of visualization applications, e.g., by hiding the details of graphics APIs and shader programming.

Comparable to the approaches mentioned before, this thesis develops a visualization pipeline that implements the data-centered visualization process by means of the GPU. That is, all steps of the visualization process, i.e., preprocessing, filtering, mapping, and rendering, are implemented using shader programs. Many of the aforementioned approaches can be applied and integrated into this GPU-based visualization pipeline. It uses an efficient data structure for storing multivariate data on the GPU and applies generative geometry to create the geometric representations from the input data. This is demonstrated on two different types of representations, raster-based density maps, as well as direct geometric representations (lines, tubes, ribbons), which can also be combined.

Chapter 3

Concept of a Framework for Visual Analytics of Geo-Temporal Data

In this work, a software framework for geo-temporal visualization techniques is developed. It includes GPU-based techniques for data management and rendering that facilitate real-time processing and visualization of large geo-temporal data sets, as well as interactive filtering and mapping. The software architecture of the framework has been designed to support the development of specialized solutions such as decision support systems and scientific analysis applications, and its use is demonstrated on domain-specific visualization methods, in particular the visualization of attributed 3D movement trajectories, 2D density maps, and large geo-referenced networks.

In general, visual analytics applications and systems tend to be highly specialized with regard to the application areas and types of data that are being analyzed. The choices of visualization methods, analysis tools, and interaction methods largely depend on the specific use case, data sets, and analysis tasks at hand, and their specific combination and adaptation to the task is very important for developing effective solutions for a given use case. Therefore, visual analytics systems are often tailored towards specific application domains and use cases.

From a technical point of view, however, visual analytics systems often share similar challenges and requirements. The visualization of large data sets in combination with interactive exploration requires not only fast rendering approaches, but also data management techniques that enable fast access to the underlying data, for example, to implement interactive filtering, selection, and analysis functions. In particular, the interactive configuration of visualization components requires rendering techniques that can be parameterized as fast a possible, i.e., that do not need costly recalculations or regeneration of large geometries. In these areas, generic components can be developed that are applicable in many visual analytics scenarios, although the specific use cases may be very different.

This chapter contains the concept of the software framework. First, requirements (Section 3.1) for the visualization of large spatio-temporal data sets are identified, and challenges (Section 3.2) for a GPU-based processing and visualization framework are derived. Finally, the concept and architecture for the GPU-based visualization framework are presented (Section 3.3).

3.1 Requirements for Interactive Visualization of Spatio-Temporal Data

In this section, the requirements for a visual analytics framework for geo-temporal data are explained based on the data characteristics as described in Section 1.1.

R.1 Visualization of Geospatial Data The main purpose of this framework is to enable analysis of geo-temporal data. Therefore, it should represent spatial data in a way that allows users to perceive and interpret spatial information such as positions and extents from the displayed data. This includes a visualization of individual items as well as clustered or aggregated data over space and time. To facilitate the interpretation of spatial data, it should also be able to embed such data into a visualization of the geographic context, for example an interactive map or globe.

R.2 Visualization of Temporal Data Users should also be able to assess temporal information contained in the data, such as the temporal relationships between objects, the order of events, the movements of objects, or the evolution of objects over time. Therefore, the framework should support the visualization of temporal information, for example using direct depictions of time, indirect approaches such as animation and filtering, and aggregation of data over time.

R.3 Visualization of 3D Data In many cases, geospatial data is gathered and depicted as being two-dimensional (i.e., latitude and longitude). This is sufficient when only the positions on earth are relevant, but not their height. Some data however is inherently three-dimensional. For example, for an observation of actual aircraft movements, rather than their connections between airports, the 3D information is inherently important. Also other research areas such as the analysis of atmospheric data in climate research, produce truly 3D geo-data. The visualization framework should therefore be able to represent geospatial data in both 2D and 3D contexts, depending on the type of data that is available, as well as user configuration. When 3D data is important for the analysis, it should be depicted in detail using an appropriate virtual 3D environment. However, a user should also be able to reduce 3D data to 2D representations whenever that is sufficient for the current analysis task in order to minimize the perceptual problems inherent in 3D visualizations and to relieve the visualization from as much visual overhead as possible.

R.4 Visualization of Multidimensional Data In addition to depicting geo-temporal data, the framework should also support the visualization of attached data attributes. For example, movement trajectories may contain attributes such as the current velocity and acceleration at each subsequent position, or sensors in climate research may gather information such as temperature or humidity over time.

An analysis of such attributed data includes several aspects: accessing the actual attribute values at specific positions and points in time, observing the development of attribute values in both spatial and temporal dimensions, finding correlations

between attribute values and the spatial surrounding, as well as finding correlations between different attributes to each other. A visualization system, therefore, needs to depict attribute values in a way that allows users to determine attribute values and trends visually, to identify extreme values, e.g., outliers, and, if possible, to visualize and compare attributes simultaneously.

R.5 Visualization of Large Data Sets Due to the many data dimensions, geo-temporal data tends to become very large. Even a moderate number of attributes per data item, multiplied by the spatial resolution and again multiplied by the temporal resolution of a time series, easily results in "big data". For the detailed analysis of spatio-temporal phenomena, however, dense spatial and temporal resolutions are also preferable.

In the context of visual analytics, a pre-filtering or aggregation of the data to reduce the number of items for visualization and analysis, is not always desirable, as it involves a pre-selection of data items and, therefore, adds a bias to the data. Thus, the visualization system needs to support the visualization of large data sets, using appropriate techniques for swapping and reloading data according to the interaction with the user, and should provide the user with tools for aggregation and filtering, rather than applying them automatically.

R.6 Interactive Exploration and Analysis For effective analysis, users need to visualize, combine, and correlate different aspects of the data. In an interactive process, they can thereby examine data values in various situations, recognize trends and outliers in the data, to test out and verify their hypotheses on the data. To enable such interactive analysis, the visualization system should be highly configurable. It needs to support interactive spatial and temporal filtering to reduce the amount of displayed data, a configurable mapping to depict different aspects of the data, as well as the integration of automatic analysis results into the interactive system. To cope with the diversity of the data, the framework should also support the combination of different visualization techniques, either directly in a single visualization or indirectly in the sense of linked views [151].

3.2 Challenges for GPU-Based Implementations

In the previous section, the requirements for an interactive visualization system for spatio-temporal data have been identified from a user perspective. Based on these requirements, this section summarizes the challenges that occur for the implementation of such a system. This includes both, conceptual challenges arising from the nature of the intended visualization framework, such as 3D visualization, as well as technical challenge for the implementation of the desired techniques on GPU-based architectures.

C.1 Memory Consumption To enable a visualization of large spatio-temporal data sets, the memory consumption for the data and its geometric representations on the GPU should be minimized. Since the selection of the subset of data that is visualized happens interactively, a pre-selection and preprocessing of the data, which would limit the amount of data needed to be uploaded on the GPU, is not possible. Therefore, as much data as possible should be loaded directly onto the GPU, since updates between CPU and GPU memory, or even loading data from disk, would increase the loading time for the data and therefore hinder the interactive process. However, large data sets combined with complex geometric representations needed for the visualization lead to high memory demands. Therefore, both the representation of the input data itself, and its geometric representations should be minimized as much as possible.

C.2 Rendering Performance To enable the visualization of large data sets at interactive frame rates, the applied rendering techniques must be selected and optimized for a high performance. For example, algorithms with a high runtime complexity should be avoided, as large numbers of items need to be processed. Optimization strategies such as spatial data structures or CPU-based sorting and culling steps, which are usually applied for more static scenes to optimize rendering performance, cannot be as easily applied, due to the flexibility of the visualization process based on user interaction, which can affect the filtering of items as well as their positions and geometric representations in real-time.

C.3 Update Performance During the interactive exploration of spatio-temporal data sets, users must be able to select and filter the subsets of data they are interested in. This involves filtering by spatial position, by time, or by attribute values. Also, they may adjust visualization parameters to find the information they seek in the data. A modification of mapping or visualization parameters can require the modification or generation of new geometric representations, which then need to be transferred to the GPU. Filtering may also involve data to be swapped or reloaded from disk. To maintain interactive frame rates for these operations, update costs between CPU and GPU memory must be as small as possible.

C.4 Flexible Geometry Generation In the context of interactive analytics systems, a visualization component must be highly configurable to enable users to visualize, inspect, and correlate different aspects of the data. This involves the modification of mapping parameters, that control how data items are mapped to visual representations, as well as the configuration of visualization and rendering options. It can also mean the selection or combination of entirely different visualization techniques for the currently selected data. Such changes to the visualization options usually require that geometry must be updated or completely regenerated. Therefore, the visualization system must be able to flexibly generate geometry needed by the selected visualization techniques and apply the selected mapping and rendering options, while maintaining interactive frame rates.

C.5 Visual Clutter and Occlusion in 3D Visualization When using perspective projections of 3D scenes, several perceptual problems arise. Due to perspective foreshortening [138, 184], spatial positions appear distorted in the image, making perception of height and size prone to misinterpretation. The effect is inevitable, in particular when using non-stereoscopic displays, but must be accounted for, since it can disturb the cognition of the displayed data. Another major problem for 3D visualization with large amounts of visual items is posed by occlusion [58]. It is caused by overlapping items, occluding the objects behind them. A simultaneous display of large numbers of visual items also leads to the effect of visual clutter [154], which affects the ability to recognize and identify individual items.

C.6 Representation of Temporal Information For the task of analyzing spatio-temporal data, and in particular movement data, spatial data aspects (e.g., the path of a movement) and temporal data aspects (e.g., start time, duration, or the order of movements) are equally important. Therefore, a visual analytics tool for spatio-temporal data must represent both, the spatial and the temporal aspects of the data, and provide methods for exploring and finding correlations between them. However, to visualize such temporal information together with spatial data poses a big challenge, since visualization of spatial data is already demanding, and there is no natural metaphor for visualizing time.

A direct method for visualizing temporal data is to map time to a visual variable. For example, time can be expressed by the height of visual items, however, this method is not applicable if the data itself is three-dimensional. Using other visual variables, such as color, for expressing time is difficult to read and interpret. Indirect methods convey time by other means, e.g., by providing multiple images, one for each point in time, by animation, or by interactive exploration. The best choice for representing temporal data depends on the specific use case and analysis task.

3.3 Framework Concept and Architecture

Based on the requirements and challenges identified before, this section presents a concept for a framework for visual analytics of geo-temporal data. It provides generic components for processing and visualizing massive geo-temporal data, which can be applied and combined to implement interactive visualization systems.

The framework is centered around a GPU-based visualization pipeline, which implements all central steps of the pipeline on the GPU. Input data is uploaded directly to the GPU, and data processing, filtering, geometry generation, and mapping are all executed on the GPU as part of the rendering. This provides a maximum of flexibility, as all steps can be configured in real-time with minimal update costs. For example, filtering and mapping options can be updated interactively, and even large changes like selecting the type of geometry can be performed in real-time, as no preprocessing on the CPU or uploading of new data are required. On the other hand, overall rendering performance is decreased, since the entire visualization pipeline needs to be executed for each frame.

3.3.1 GPU-Based Visualization Pipeline

Visualization systems can be described in terms of the visualization pipeline concept described by Haber [71], which identifies the fundamental steps of the visualization process. According to Haber, the visualization pipeline can be divided into three fundamental steps: preprocessing, mapping, and filtering.

In the first step, input data is loaded and processed, converting it into a form that is suitable for the further steps of the pipeline. This usually involves a preprocessing of the data, e.g., for converting data types, filling in missing data attributes, as well as checking and if possible correcting the validity of data values. Also, filtering is often performed as part of this process, reducing the data to a subset for the visualization.

The mapping step determines how input data is represented in the visualization. This involves the creation of actual geometry, as needed by the chosen visualization method, and the mapping of properties of the input data to visual variables of the generated geometry. The mapping step therefore defines how input data is transformed into visual artifacts.

In the final step, the generated geometry is rendered onto the screen, creating an actual image as a result. The generated geometry thereby provides the configuration for the rendering technique, which may include complex rendering processes such as lighting, shading, as well as other graphical and post-processing effects.

Current visualization systems typically make use of modern graphics hardware, by applying hardware accelerated methods for processing and rendering. However, following traditional rendering methods, this is often implemented in such a way that mainly the rendering step is executed on the GPU, while the complex geometry generation process is handled on the CPU side. As a result, preprocessing, filtering, and mapping are seldom implemented on the GPU side.

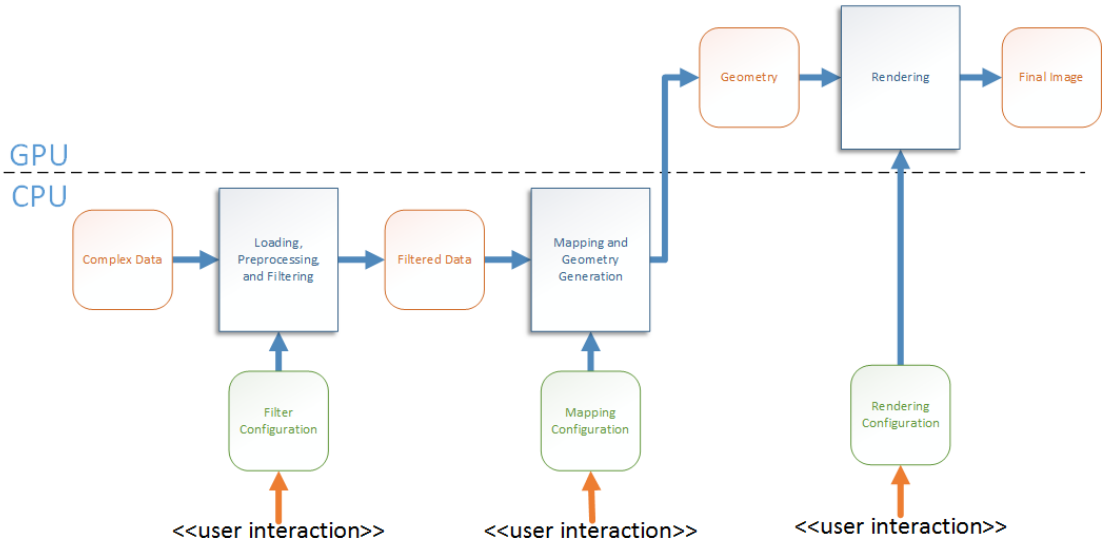


Figure 3.1: *Visualization pipeline.*

Figure 3.1 explains this on the example of a typical GPU-accelerated visualization pipeline. While rendering is implemented on the GPU, filtering, mapping, and geometry generation are mainly executed on the CPU, resulting in the need to upload data from CPU to GPU memory whenever the geometry has changed. As indicated, user interaction processes, for example the alteration of filtering and mapping configurations, result in the re-generation of at least part of the geometry, and therefore trigger an upload of data from CPU to GPU memory. This can require longer time periods for updating the visualization artifacts in response to user interaction and therefore has an impact on the interactivity of the system as a whole.

As an alternative, this work defines a visualization pipeline, in which all three stages (filtering, mapping, and rendering) are implemented on the GPU. In particular, complex input data is uploaded directly to the GPU, using an optimized data structure called an *attributed vertex cloud*. Instead of creating complex geometry on the CPU and uploading it onto the GPU for rendering, mapping and geometry creation are deferred and executed on the GPU during rendering. This reduces the memory consumption of the data and enables interactive configuration of filtering, mapping, and geometry creation at runtime. The concept of this rendering pipeline is explained in Figure 3.2.

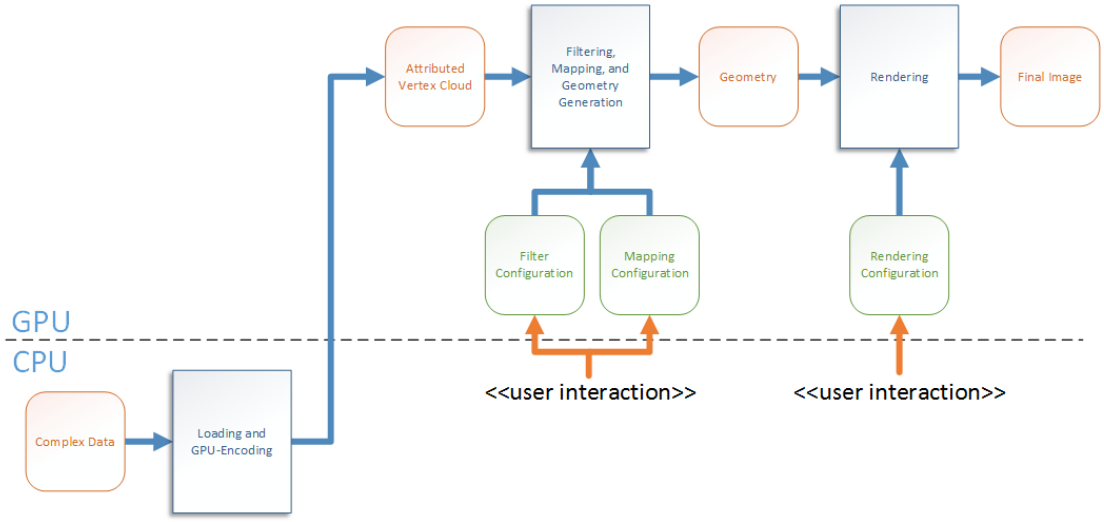


Figure 3.2: *GPU-based visualization pipeline.*

This kind of visualization pipeline has some advantages with regard to the defined challenges. Due to the smaller size of attributed vertex clouds in comparison to the actual generated geometries, memory consumption (C.1) is reduced. By generating the geometry on the fly, using data that is already present on the GPU, a costly regeneration and upload of geometry to the GPU are avoided. During user interaction, such as the modification of filtering and mapping configurations, only the configuration buffers need to be updated on the GPU, therefore, update costs (C.3) are much lower. Rendering performance (C.2) largely depends on the complexity of the generated geometry. The main performance gain of attributed vertex clouds is generated by rendering the whole

geometry with a single draw call, on the other hand, the dynamic generation of geometry during rendering also makes the rendering process more complex. In summary, therefore, rendering performance is not largely influenced by the GPU-based pipeline. A detailed performance analysis is performed in Chapter 7.1.

3.3.2 Attributed Vertex Clouds

To represent 3D scenes in interactive computer graphics applications, polygonal geometries, such as triangle meshes, are used. They are stored in a way that is optimized for the GPU-based rendering pipeline:

- **Vertex buffers** contain the positions of vertices and corresponding data such as texture coordinates, texture and material indices, normal vectors, IDs, etc.
- **Index buffers** are used to define geometric primitives that are rendered, e.g., triangles or triangle strips. They reference vertices by index and are therefore capable of reusing vertices that would otherwise need to be duplicated.
- Additional **data buffers** are used to store data referenced by the geometry, for example, textures, normal maps, or shadow maps.

Using this method, meshes of arbitrary complexity can be represented and rendered. However, while even complex meshes can be rendered efficiently on the GPU, they also have relatively large memory requirements. Also, a dynamic modification of geometries as part of animations or in response to user interaction is costly, as it requires the geometry buffers to be updated from CPU to GPU memory. Therefore, it is usually avoided to update large complex geometries in real-time. Instead, geometric transformations (e.g., skeletal animations), parameterized instancing, and the replacement of corresponding data (e.g., materials or textures) are applied to modify the appearance of geometries during rendering and to make scenes flexible without modifying the geometry itself.

While the above applies to complex geometry in general, there are several types of simpler geometries (e.g., cuboids, spheres, tubes, or splats), which can be generated procedurally. In the described use case for visual analytics systems, scenes often consist of large numbers of such simple geometries. They represent individual data items of a displayed data set and are parameterized individually to map properties of the input data to visual variables. For interactive analysis and exploration systems, the displayed geometries need to be influenced by user interaction, for example to enable interactive mapping, filtering, and highlighting.

For such use cases, another approach for representing and rendering geometries, called *Attributed Vertex Clouds*, has been developed. The general concept is explained in Figure 3.3. An attributed vertex cloud contains a list of vertices, in which each vertex corresponds to one instance of the target geometry. Depending on the complexity of the use case and analysis task, such a vertex can either describe the target geometry directly, or represent a general data point of the application domain, e.g., a tracking point of an attributed movement trajectory.

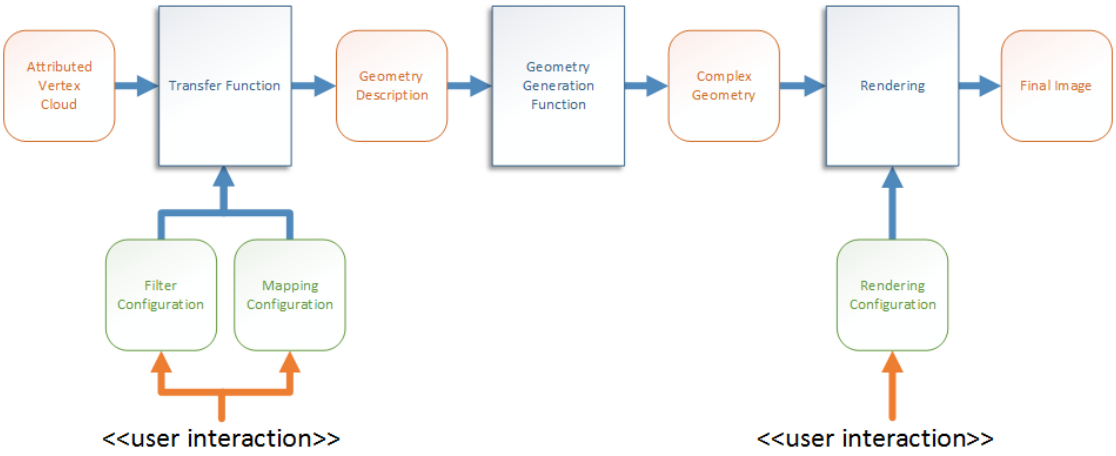


Figure 3.3: *General rendering concept using Attributed Vertex Clouds.*

During rendering, each vertex is processed and a *transfer function* is applied, which converts the vertex into a parameterized description of the target geometry. For example, a cube can be defined by its center point (x, y, z), the length of its edges ($size$), and a color. The transfer function determines the type and configuration of the target geometry based on the input vertex, but it can also take into account other parameters such as global configurations, the current camera position, or interaction states (e.g., selection).

Finally, a *geometry generation function* is invoked, which generates the actual geometry. It uses a geometric template or a procedural representation of a target geometry, applying the parameters selected by the transfer function. Then, that geometry is rasterized and rendered to the screen.

3.3.3 Interactive Attribute Mapping

Depending on the use case and analysis task, the complexity of the transfer function can vary strongly. It can range from a simple value mapping to a function that represents a complex decision tree for selecting the right visual output for each data item. It can therefore be used to implement a lot of different visualization and interaction metaphors.

Mapping: As its core functionality, the transfer function defines the mapping between input data attributes and visual properties of the output geometry. For example an input attribute can be mapped to the size or color of the output geometry. Such a transfer function will usually involve normalization of input values, an application of an interpolation function, to define the mapping between input and output value.

Dynamic Geometry: A more complex transfer function can not only define a static mapping between data attributes and geometric properties, but it may select entirely different geometries and mappings based on the properties of the data item. This can be applied to differentiate between different classes of items, to highlight specific elements, or to respond to interaction events, such as selection.

Dynamic Positioning: Since the input positions in the attributed vertex cloud do not necessarily correspond already to the final 3D positions in the visualization, the transfer function is also responsible for calculating the final positions of the output geometries for each data item. This enables for example dynamically switching between 2D and 3D visualizations, or applying different map projections for geographic data visualization. In the case of general information visualization, data items usually do not contain an inherent position at all. Therefore, the visualization process involves a spatialization step, in which a layouting algorithm calculates the positions for all items. Depending on the complexity of that specific layout algorithm, this process can also be implemented as part of the transfer function, allowing layout algorithms to be switched dynamically.

Level-of-Detail: By taking into account the distance of the virtual camera to the displayed item, level-of-detail methods can be implement in the transfer function. For example, by reducing the tessellation level of the generated geometry for large distant objects, rendering performance can be optimized.

Level-of-Abstraction: Using the same technique, the transfer function can be used to apply level-of-abstraction techniques. Different geometric types and mapping configurations can be selected, for example to display detailed geometry for highlighted and selected items, while presenting only basic information for all other items.

Filtering: Filtering can be performed in the transfer function by selecting empty output geometry for data that matches the filter criteria. In that case, the geometry generation function will emit no geometry, so the data element is effectively filtered.

To give an example for applying these techniques, a complex transfer function for the interactive exploration of spatio-temporal data sets is developed in the following. It enables users to visually classify the input data by defining the subsets of data they are interested in, defined for example by certain attribute values, and assigning unique visual styles for each subset. Using interactive spatial and temporal filtering, they can identify the data subsets and draw conclusions as to the frequency of occurrence and the spatial and temporal distributions. They can then redefine the classifications to differentiate data clusters further, or use the attribute mapping options to gain a more detailed look into the data set.

To implement this mapping concept in an interactive system, an additional data structure is needed to encode the visual styles for each of the data subsets. This *visual configuration* controls which visual property is assigned what value, either by defining a static value for it or by selecting a data item, which is then mapped onto the visual property. These configuration can be defined by the user, using for example a simple UI or a scripting language, and are uploaded to the GPU into a *configuration buffer*.

As a second part, the criteria for selecting a configuration for a data item must be defined. In theory, an arbitrary complex function could be defined here to select the configuration based on the values of the data item and the current interaction states (e.g.,

filtering, and selection). For simplicity, a less flexible approach is used in this example: the user selects a *primary attribute*, which defines the index of the configuration that is to be used. In addition to that, for each configuration one of four sub-configurations is selected, defining three levels-of-details based on the current camera distance, and one configuration for the case that the item is currently selected by the user.

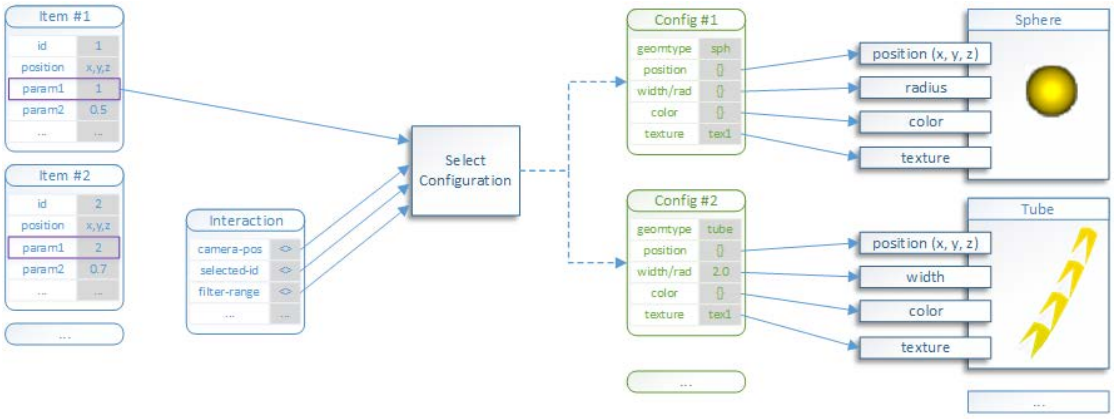


Figure 3.4: *The first step of the transfer function selects the visual configuration for the current data item, based on the data attributes of the item and current interaction states.*

Consequently, the transfer function for this visualization is split up into two parts. First, the visual configuration for the current data item is selected based on the primary data attribute and the current interaction states (Figure 3.4). After the visual configuration has been selected, it is read from the buffer and applied to the target geometry description, thereby also defining the mapping from input values to visual properties (Figure 3.5). Then, the geometry is finally generated and rendered to the screen.

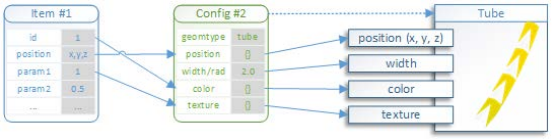


Figure 3.5: *The second step of the transfer function applies the mapping between data attributes and visual properties as defined by the selected visual configuration.*

Chapter 4

OpenGL-Based Framework Implementation

For creating an implementation of the described concepts, the programmable rendering pipeline as available on current computer graphics hardware can be used. In general, it should be feasible to implement the concept with any of the major graphics APIs, such as OpenGL [166], OpenGL ES [137], DirectX [128], or Vulkan [163]. However, a restriction must be made with regard to the selected version of the rendering API.

Most notably, dynamic geometry generation must be available. As the simplest approach, geometry shaders [167] can be used. They are supported in OpenGL since version 3.2, and DirectX since version 11. As of the time of writing, OpenGL ES does not support geometry shaders. As an alternative, the dynamic creation of geometry could also be implemented using compute shaders [168] that generate dynamic vertex buffers as their output. However, this is a bit more complicated, as compute shaders are not integrated directly into the rendering pipeline and therefore need to be executed separately. On the other hand, this also provides more flexibility in controlling and optimizing the geometry generation. Compute shaders are available in OpenGL since version 4.3, OpenGL ES since version 3.1, and DirectX since version 11.

In this chapter, an implementation of the GPU-based visualization pipeline is presented based on OpenGL 3.3.

4.1 Architecture

In Figure 4.1, the main components of the implemented visualization framework are presented. The implementation is based on OpenGL 3.3 and a slim computer graphics middleware layer is used, which manages the general application and rendering logic and provides an object oriented abstraction for OpenGL.

In general, a more complex scene graph system or game engine could have been applied to implement the visualization framework. However, such systems are usually designed and optimized for a specific class of use cases, thus making assumptions about the structure of the displayed scene and its rendering pipeline.

For example, most real-time rendering systems are optimized towards the rendering of large, dynamic scenes, which are composed of more or less static, complex 3D triangle meshes. Rendering performance is optimized for example by analyzing the scene before

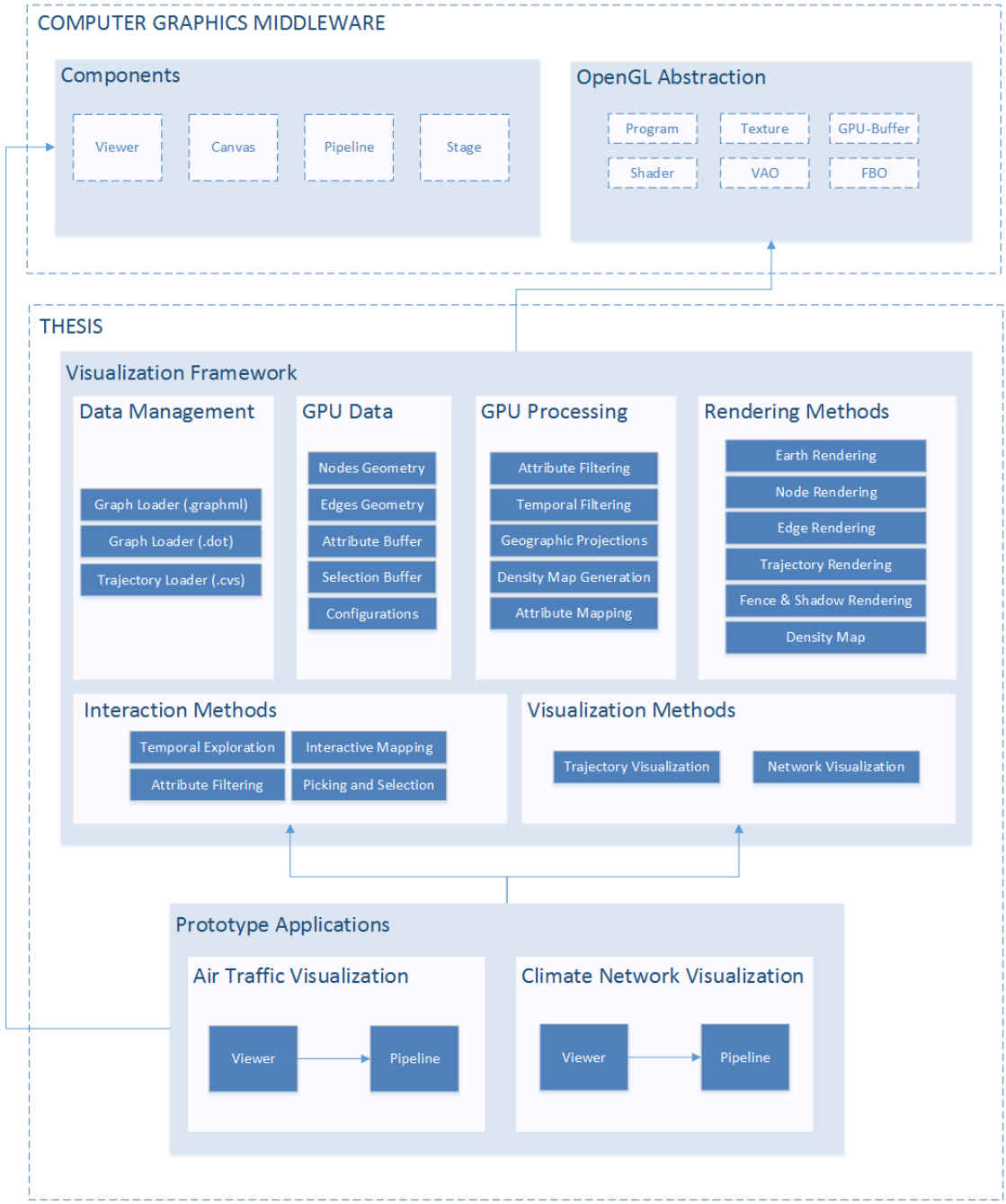


Figure 4.1: *Software architecture of the developed visualization framework.*

rendering, typically using a culling pass, which identifies all objects included in the current view frustum, thus rendering only those objects that are currently visible. For larger scenes, spatial data structures are often applied. Additional methods, such as sorting the scene to minimize state changes or context switches, are used to further optimize the rendering performance.

While such systems are highly optimized towards their intended use case, they can hinder the implementation of different rendering approaches. The framework concept as described in this thesis, for example, depends on processing large numbers of data items on the GPU by encoding them as attributed vertex clouds (see Chapter 3.3.2). In a sense, they encode an entire scene by means of a single GPU geometry, and do therefore not correspond well to the object scene model as described before. For each data item in a vertex cloud, the visibility, position, and shape of its corresponding visual geometry are generated dynamically during rendering and cannot be determined before. They also heavily depend on user interaction and can change drastically from frame to frame. Thus, a culling logic as described before, which determines the visibility of an entire geometry based on the position of a virtual object in the scene, would fail and therefore introduce unnecessary overhead. On the other hand, restructuring the rendering approach to conform to the assumptions of the rendering engine would destroy the flexibility and performance of the proposed rendering approach.

For such unusual rendering approaches, many of the assumed optimization steps will therefore not work any more. Similar problems can occur with regard to the GPU-based data structures, which need to be maintained and updated dynamically for visualization and interaction purposes. Implementing such methods in terms of the logic of a given rendering engine can prove difficult.

Therefore, a computer graphics middleware has been selected that abstracts the most important components needed for computer graphics applications, but does not impose a specific scene structure or make any assumptions about the rendering process otherwise. The rendering process is defined using a pipeline concept, which helps organize rendering into individual, reusable steps, but without introducing automatic processes or restrictions to the rendering process itself.

4.2 Attributed Vertex Clouds

To represent the multidimensional input data, attributed vertex clouds (AVCs) need to be stored on the GPU. In theory, AVCs could be represented by vertex buffers, using one vertex attribute per data attribute. According to the specifications, current graphics cards should support at least 16 or 32 attributes per vertex, which would be sufficient for many data sets.

In practice, however, the number of vertex attributes is often very limited, and there are many restrictions as to the combination of vertex attributes and their corresponding vertex attribute bindings. During the development of this implementation, some graphics card and driver combinations even exhibited non standard-conforming behavior when using large numbers of vertex attributes, making it nearly unpredictable whether the

implementation would be supported on any given system. Also, using large numbers of attributes per vertex has a significant impact on the rendering performance, as the GPU has to fetch all data attributes per vertex, even if they are not used later.

Therefore, instead of storing AVCs directly as vertex attributes, the data is split up into two parts: a vertex buffer that stores the geometry (e.g., points or lines), and a separate attribute storage containing the data attributes for each item. The vertex array now consists of only two *vec3* vertex attributes. In the first, the position of the vertex is stored, e.g., (x, y, z), or $(lat, lon, height)$. The second vertex attribute contains the IDs of the data item (e.g., a node ID and a line ID), which are used to look up the data attributes of the item, and a time stamp for spatio-temporal data sets. These are the components that are most likely to be needed for each vertex. For example, when spatial or temporal filtering is applied, many vertices can be rejected very early in the rendering process, so the other data attributes do not need to be accessed any more. Therefore, unnecessary texture lookups are avoided if the vertex is already filtered due to its position or time stamp.

4.3 Attribute Storage

Data attributes are stored in a separate GPU buffer and accessed using a buffer texture object (GL_TEXTURE_BUFFER). This buffer concept allows us to access the attribute data as a large one-dimensional floating-point texture without interpolation or filtering by default. Buffer textures [169] also have very few size limits as opposed to other OpenGL textures and GPU storage types, therefore they are suitable for storing large multidimensional data sets.

When a data set is loaded, it is linearized into the buffer, assigning each item a sequential ID, which allows random access to all data items and their attributes. In addition to that, a data analysis takes places that calculates the minimum, maximum, and mean values for each data attribute. This information is also stored in the attribute storage at predefined locations, so that it can be accessed later in a shader program. This can be used for example by the transfer function to implement different interpolation functions for attributes and allow users to switch between them interactively.

4.4 Visual Configurations

To enable the implementation of dynamic attribute mapping and geometry generation in shader programs, the visual configurations, which control the visual style and attribute mapping, must also be represented on the GPU. For this, uniform buffers are used. They can be accessed easily from within shader programs, and for most visualization methods, only a small number of visual configurations are used, so they fit into the size limit of uniform buffers. In cases where larger numbers of visual configurations are required, they could also be stored using buffer textures.

As the purpose of visual configurations is to define the parameters of the visual representations and rendering options, a configuration provides a field for every parameter of the actual rendering method. For example, a configuration may define the output geometry type ('line', 'tube', or 'sphere'), output size, output color, and tessellation level. All of these attributes can either be assigned a static value or contain a reference to a data attribute. To enable this, a simple encoding is applied: values greater or equal to zero are interpreted as static values, while numbers smaller than zero encode the index of the mapped attribute. This encoding is sufficient, as attribute indices can never be negative and most rendering options also only use positive numbers. More complex encodings can be defined in cases where for example negative values must be available for some rendering options.

Listing 4.1 shows how visual configurations can be stored in a uniform buffer and be accessed from within a shader program. In this example, a maximum number of 16 configurations can be defined, each of which can further be specialized for 4 levels of detail. Given a configuration ID and LOD level, the index to the specific configuration can be easily calculated as exemplified in function getConfigID.

```
1  #define MAX_CONFIGURATIONS  16
2  #define NUM_LOD              4
3
4  struct Configuration {
5    float geometryType; // 0 = discard, 1 = tubes, 2 = sphere
6    float positionX;    // Value or attribute mapped to position.x
7    float positionY;    // Value or attribute mapped to position.y
8    float positionZ;    // Value or attribute mapped to position.y
9    float radius;       // Value or attribute mapped to tube/sphere radius
10   float color;        // Value or attribute mapped to color
11   float textureID;    // Texture ID
12   float colorMapID;   // Color map ID
13   float tessellation; // Tessellation level
14   // ...
15 };
16
17 layout(std140) uniform CONFIG
18 {
19   Configuration confs[MAX_CONFIGURATIONS];
20 };
21
22 int getConfigID(int classID, int lod)
23 {
24   return classID * NUM_LOD + lod;
25 }
```

Listing 4.1: *Declaration of visual configurations and accessing them from shader programs.*

4.5 Rendering

To trigger the rendering of an attributed vertex cloud, a single draw call is issued that draws the entire vertex array in a single step (*glDrawArrays*). For each vertex, this triggers the transfer and geometry generation functions, which are implemented as a combination of vertex and geometry shaders. Inside these shaders, filtering, mapping, and geometry generation take place. If a data item is not filtered, the geometry shader will produce an output geometry, which is then rasterized into the final image. For filtered data items, no output geometry is generated, so the rendering pipeline automatically stops after the geometry shader.

Although vertex buffers for AVCs tend to be very large, having a single draw call significantly improves rendering performance as compared to multiple draw calls per geometry, instanced drawing, or even indexed geometry. Graphics cards are highly optimized for processing large numbers of vertices in parallel, so the number of vertices does not impact rendering performance very much, as long as no geometry is generated by it. Many vertices are immediately omitted, for example because they are filtered by the transfer function, so no geometry is created for them. In the end, the actual number and size of the created geometries has the biggest impact on rendering performance.

4.6 Transfer Function

An implementation of a transfer function involves fetching one vertex, looking up its attributes, and choosing the visual configuration based on the data item and current interaction states. Since it is invoked exactly once for each individual vertex, a transfer function could easily be implemented by means of a vertex shader. However, analogous to the storage of vertex attributes in attribute buffers rather than vertex attributes, it should be avoided to pass unnecessary data between vertex and geometry shader. If for example a vertex is filtered by the transfer function, it is unnecessary to pass the entire visual configuration to the geometry shader just to omit the vertex right after. Therefore, an implementation of the transfer function as part of the geometry shader can be advantageous. In that case, the vertex shader can be implemented as a simple pass-through for the vertex attributes.

Listing 4.2 contains the implementation of a transfer function as part of a GLSL shader program. First, it applies filtering, beginning with the information that is instantly available through vertex attributes and thus avoiding unnecessary data fetches. If the vertex is not filtered, the function then fetches the primary attribute value of the vertex to determine the configuration ID that is to be used. Then it calculates the output position of the vertex according to the chosen configuration and determines the level-of-detail, based on the camera distance and interaction states. With that information, the final configuration ID is determined and the position of the respective configuration in the configuration buffer is calculated.

```
#version 330

// Input: Position (lat / lon / height) in radians
layout (location = 0) in vec4 v_position;

// Input: Node-ID / Line-ID / Time / -
layout (location = 1) in vec4 v_attrs;

// Options
uniform vec2  idFilter;         // ID filter (min, max)
uniform vec2  timeFilter;       // Time filter (min, max)
uniform int   primaryAttribute; // Index of primary attribute
uniform float selectedId;       // ID of currently selected item

void main()
{
    // Fetch vertex data
    float node = v_attrs.x;
    float line = v_attrs.y;
    float time = v_attrs.z;

    // Perform filtering: by ID, by time, by attribute value
    bool visible = true;
    visible = visible && (line < idFilter.x || line > idFilter.y));
    visible = visible && (time < timeFilter.x || time > timeFilter.y));
    visible = visible && (filterAttrs(node, line) > 0.0);

    // Determine class ID by looking at the primary attribute
    int class = attributeValue(node, line, primaryAttribute));

    // Calculate position of the vertex (ignoring LOD for now)
    int config = class * 4;
    vec4 position = calculateVertexPosition(
        v_position, node, line, config
    );

    // Choose level of detail ('lod-3' for selected items)
    int lod;
    if (line == selectedId) {
        lod = 3;
    } else {
        lod = getLOD(getLinearDepth(viewMatrix * position);
    }

    // Determine final configuration ID
    config = class * 4 + lod;

    // Perform attribute mapping
    // ...
}
```

Listing 4.2: *Implementation of a transfer function in a GLSL shader.*

4.7 Attribute Mapping

After the transfer function has determined which visual configuration is applied for a given vertex, the actual attribute mapping must take place. This is demonstrated in Listing 4.3. First, the selected visual configuration is accessed by index in the uniform buffer. For each output value, the mapping configuration is looked up from the configuration, providing either a static value or referencing a data attribute, which is then fetched from the attribute storage buffer. This basically implements the encoding for attribute mappings as defined in 4.4. In the end, a configuration that describes all necessary parameters for the output geometry and its rendering has been created. This result is passed on as input to the geometry generation function.

```
1  #version 330
2
3  // Get config value for vertex <index>
4  //   if input >= 0, the value itself is the config value
5  //   if input <  0, an attribute value is mapped
6  float confVal(int node, int line, float input)
7  {
8    if (input >= 0.0) {
9      return input;
10   } else {
11     return attributeValue(node, line, int(-input) - 1);
12   }
13 }
14
15 void main()
16 {
17   // ...
18
19   // Perform attribute mapping
20   float geometryType = confVal(node, line, confs[config].geometryType);
21   float radius       = confVal(node, line, confs[config].radius);
22   float color        = confVal(node, line, confs[config].color);
23   float textureID    = confVal(node, line, confs[config].textureID);
24   float colorMapID   = confVal(node, line, confs[config].colorMapID);
25   float tesselation  = confVal(node, line, confs[config].tesselation);
26
27   // Generate dynamic geometry
28   // ...
29 }
```

Listing 4.3: *Implementation of attribute mapping in a GLSL shader.*

4.8 Geometry Generation Function

As the last step of the pipeline, the final geometry has to be generated by the geometry shader. The necessary parameters for generating that geometry have been defined by the attribute mapping step, so this stage selects the type of geometry that is going to be generated and invokes the corresponding geometry generation function (Listing 4.4).

```
#version 330

void main()
{
  // ...

  // Generate dynamic geometry
  if (geometryType == 1) {
    generateSphere(
      position, radius, color, textureID, colorMapID
    );
  } else if (geometryType == 2) {
    generateTube(
      position, radius, color, textureID, colorMapID, tesselation
    );
  }
}
```

Listing 4.4: *Dynamic dispatch of geometry generation according to attribute mapping.*

A geometry generation function involves the implementation of a procedural geometry generation algorithm, including the generation of additional output attributes such as texture coordinates or attributes needed by further post-processing steps. As an example, Listing 4.5 demonstrates the principal functionality of a fairly simple geometry generation function, generating shaded, colored spheres. Taking the configuration for the geometry as its input, it emits the resulting attributed geometry, in this case a screen-aligned quad, which will be processed using a GPU-based raycasting method in the fragment to visually appear like a sphere [170]. This allows us to generate perfectly screen-oriented spheres at a high rendering performance.

```
#version 330
#extension GL_ARB_uniform_buffer_object : enable

layout (points) in;
layout (triangle_strip, max_vertices = 40) out;

// Geometry output
out vec3  g_pos;
out vec3  g_center;
out vec3  g_up;
out float g_radius;

```

```
13 void generateSphere(position, radius, color, textureID, colorMapID)
14 {
15     // Calculate sphere center
16     vec3 M = (viewMatrix * position).xyz;
17     vec3 u = -M;
18 
19     // Calculate billboard
20     vec3 dir = normalize(M);
21     vec3 B   = M - dir * radius;
22     vec3 dx  = cross(vec3(0.0, 1.0, 0.0), dir) * radius;
23     vec3 dy  = cross(dx, dir) * radius;
24 
25     g_pos       = B - dx + dy;
26     g_center    = M;
27     g_up        = u;
28     g_radius    = radius;
29     gl_Position = projectionMatrix * vec4(g_pos, 1.0);
30     EmitVertex();
31 
32     g_pos       = B + dx + dy;
33     g_center    = M;
34     g_up        = u;
35     g_radius    = radius;
36     gl_Position = projectionMatrix * vec4(g_pos, 1.0);
37     EmitVertex();
38 
39     g_pos       = B - dx - dy;
40     g_center    = M;
41     g_up        = u;
42     g_radius    = radius;
43     gl_Position = projectionMatrix * vec4(g_pos, 1.0);
44     EmitVertex();
45 
46     g_pos       = B + dx - dy;
47     g_center    = M;
48     g_up        = u;
49     g_radius    = radius;
50     gl_Position = projectionMatrix * vec4(g_pos, 1.0);
51     EmitVertex();
52 
53     EndPrimitive();
54 }
```

Listing 4.5: *Implementation of a geometry generation function in a GLSL geometry shader.*

Chapter 5

Air Traffic Analytics

This chapter describes an application of the developed visual analytics framework in the field of air traffic analysis, in particular, the analysis of aircraft movements over a long period of time. The main goals are to explore, visualize, and analyze historic data sets of aircraft movements in order to understand movement dynamics and derive insights from past events. The described techniques and application examples are intended as building blocks in the context of air traffic management (ATM), in particular for analyzing past air traffic, as well as improving management or planning tasks in future. They are not intended for air traffic control (ATC) systems, which are used for monitoring and controlling current air traffic and need to have efficiency and safety as their main concern.

5.1 Use Case

Air traffic management (ATM) describes the entirety of processes and systems that enable or support the air traffic system around the world, i.e., to enable aircrafts to travel from one airport to another. It includes the operational part of air traffic control (ATC), as well as broader management tasks, such as airspace management, flow and capacity management, and planning tasks.

While an important part is the monitoring and management of the current situation in air traffic, ATM is also concerned with the analysis of past data, e.g., to determine common flight routes and traffic patterns, to analyze safety related incidents, and to derive insights from traffic data in order to optimize processes and influence the planning for the future. In this context, the analysis of air traffic movements plays an important role. Exemplary questions and interests for the analysis of aircraft movements include:

A.1 Trajectory Analysis Individual flights can be analyzed by a detailed inspection of their movement trajectories. This includes the reconstruction of the actual movement (i.e., change of position over time), as well as the assessment of corresponding attributes, such as velocity, acceleration, or orientation, over the course of a trajectory. This kind of trajectory analysis can be applied to identify air traffic anomalies, such as aerial bottlenecks and hotspots, or to reconstruct individual flights, for example for accident analysis.

A.2 Flight Route Analysis By investigating the distribution of movements over time and space, areas with higher and lower traffic volumes can be identified. From this, common flight routes can be derived and analyzed. Also, the change of traffic density in certain areas can be analyzed over time. This helps identifying both spatial and temporal hot spots in air traffic and can be be used for management and planning purposes.

A.3 Pattern Discovery and Classification A comparison of individual flights to common flight routes allows for identifying similarities and differences between flights. By correlating them to other flight attributes (e.g., departing and arriving flights or aircraft types), a classification of flights can be achieved. The objective of such a classification can be the optimization of processes and regulations, or the identification of certain types of flights in historic or real-time data.

A.4 Outlier Detection A further goal is to identify typical and untypical movement behaviors. This can be used for example to detect outliers in past and present air traffic and, if possible, to correlate them to spatial or temporal circumstances.

A.5 Adherence to Safety Regulations By using additional data such as flight routes, air traffic sectors, or safety regulations, flight traffic that violates those can be identified. A goal is to find the reasons or circumstances of such violations and to simulate and assess the consequences of administrative or regulatory actions regarding air safety violations.

A.6 Comparison of Routing Scenarios For air traffic management and planning, one requirement is to compare alternative scenarios. For example, the results of changing flight routes, or the consequences of how a new air port is designed, can be examined by simulating and comparing the resulting movement trajectories and traffic volumes.

5.2 Related Work

Over the last decades, the number of flights and passengers in global air traffic have grown rapidly. Correspondingly, applications and systems have been developed to support the management and analysis of air traffic. The main use cases for interactive visualization systems are air traffic control (ATC) and air traffic management (ATM). In ATC, the main goals are to improve efficiency and safety, and to optimize work load distribution of air traffic controllers with the support of automatic analysis systems. For ATM, on the other hand, the focus lies more on understanding and optimizing air traffic by analyzing historic data. This involves the analysis of large numbers of trajectories and the extraction of information, such as common flight routes and traffic density.

Hurter et al. developed FromDaDy, an interactive tool for the visualization of large numbers of aircraft trajectories, using scatterplots and an interaction technique based on brushing and linking [87]. Based on FromDaDy, Hurter et al. presented methods

to analyze aircraft trajectories with interactive image-based information visualization techniques, combining their former work with density-based and graph-based techniques, as well as edge-bundling [80]. They also proposed a method to extract wind parameters from aircraft trajectories, which is an important information for air traffic controllers [79]. Scheepens et al. improved on the density-based approach for analyzing trajectories by using a density map for the overview and adding dynamic particle systems to visualize the direction of movements [156].

Bourgois et al. explored the use of immersive systems for the visualization of air traffic trajectories and corresponding data, such as weather conditions, in virtual reality (VR) [34]. Hurter et al. later introduced FiberClay [85], an immersive visualization system that allows users to navigate into large numbers of trajectories and interact with them by means of filtering and selection, using VR-based interaction techniques.

Temporal bundling approaches have been developed for the visualization of trajectories and dynamic graphs [83], and have been applied for the analysis of large air-traffic trajectory data sets [99, 81]. Attribute-driven edge bundling also take into consideration additional attributes in the edge bundling approach, such as direction or time, which improves its value for air traffic visualization as compared to former undirected edge bundling [146]. Hurter et al. developed an algorithm for bundled simplification of 2D and 3D trajectories, based on functional decomposition of the underlying curves [84].

Buchmueller et al. developed a visual analytics system to explore and analyze the impact of local air traffic with regard to aircraft noise impact and violations of air traffic regulations [35]. Eerland et al. proposed a data-driven method to derive flight corridors from trajectory data sets. It allows for analyzing and visualizing flight corridors under different circumstances, such as weather conditions, and supports a measure for identifying air traffic complexity [57]. Andrienko et al. used a clustering approach for trajectories combined with interactive filtering to identify and visualize summaries of aircraft traffic trajectories [11]. They also developed a conceptual framework for comparative analysis of trajectories embedded in an interactive framework, using an analytical procedure for pairwise comparison of trajectories [22].

In the context of ATC, methods and systems have been developed to support the work of air traffic controllers. Hurter et al. developed a method to generate metro-like maps for the flight routes in a sector, to improve the visualization of flight routes in the context of ATC [86]. Maggi et al. conducted a study on the impact of display design and user characteristics on animated displays of aircraft movements in air traffic control [130].

5.3 Air Traffic Data

In this section, the types and characteristics of the air traffic related data sets are described, on which the visualization and analysis methods in this work are based.

Trajectory Data A main category of data for air traffic analysis consists of tracking data from aircraft movements. Such data sets are usually generated either by *RADAR* [171] systems that monitor air traffic, or by recording *ADS-B* [2] signals

from aircrafts. With ADS-B, an aircraft periodically broadcasts its own position together with additional information, such as identification, altitude, and velocity. This data is received by ground stations to track flights around the world.

Tracking data usually consists of large numbers of sample points, each containing the position and status of a flight at a specific point in time. By grouping the consecutive sample points for each individual flight, attributed trajectories of aircraft movements are derived. The resulting data therefore consists of two important parts: *nodes* and *trajectories*.

A *node* denotes an individual sample point, while a *trajectory* denotes the movement of an aircraft by a consecutive list of nodes. Per-node attributes include a time stamp, the current geographical location, as well as the current altitude and velocity. Per-trajectory attributes include the flight identification (ICAO call-sign), type of aircraft, and meta data about the flight, such as departure and destination airports, runway designations, or planned flight routes.

Aircraft Databases Many types of information found in tracking data are represented using international codes, mainly specified by the ICAO [88]. ICAO codes are used for example to identify airports, airlines, and aircraft types. To interpret the codes, appropriate databases must be used. In this work, an aircraft type database is applied to look up additional information about the aircrafts found in the tracking data. It contains a description of the aircraft type, the number of engines, and the weight class, which can be used to roughly categorize aircrafts by their size.

Air Traffic Sectors In international air traffic, airspace is subdivided into sectors, which define distinct areas of responsibility in air traffic control. A sector is defined by a three dimensional volume of space, usually but not always rectangular in shape. In the context of this work, sectors are interesting for example to determine traffic volumes within these sectors and their development over time. This information can be used for optimizing flight routes as well as for operational tasks, such as the assignment of air traffic controllers depending on the current traffic volume.

The main data source for air traffic visualization in this work are movement trajectories of aircrafts. However, depending on the data sources, trajectory data sets can have very different characteristics. For example, flight tracking projects such as flightradar24 [1] collect data of flights worldwide. They therefore collect flight information globally, including long-distant flights, and over a long period of time.

The resolution of these data sets tends do be low, in the range of one sample point per minute. This kind of data is useful for analyzing global air traffic, such as determining the amount of connections between international airports or areas, or identifying and categorizing types of flights worldwide, but they are not suitable for analyzing the actual movement of individual aircrafts, especially during takeoff or landing.

For analyzing flight traffic on a local or regional level, higher resolution data sets are required. For example, to determine common flight routes of approaching and departing

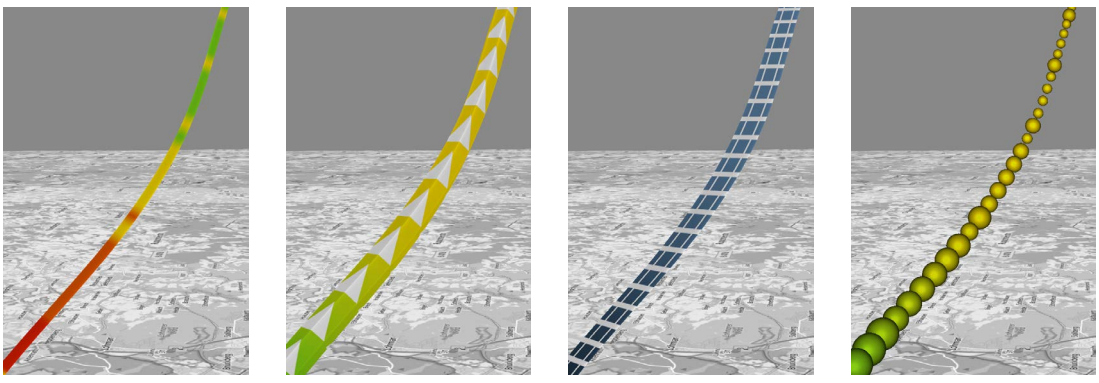
aircrafts around airports, or to assess traffic volumes in areas of high population, such as major cities, temporal resolutions of 10 seconds or less are desirable.

The main data set used in this work comprises 3D trajectories of aircraft movements. It originates from RADAR tracking in the vicinity of an airport and includes departing and arriving IFR (instrument flight rules) flights. The tracking data has a temporal resolution of four seconds and a range of about 50-70 km. A single trajectory has a duration of 5-10 min with approximately 75-150 sample points. For a duration of one month, the data set consists of a total number of 12,500 trajectories and more than 1,500,000 sample points.

5.4 Attributed Trajectory Visualization

As a fundamental component for analyzing aircraft movement data, a direct visualization method for attributed 3D trajectories has been implemented. In this approach, movement trajectories are represented by polygonal geometry, depicting the exact movements in 3D space (R.1, R.2, R.3). Several geometric primitives - lines, tubes, ribbons, and spheres - are available for the geometric representation of a trajectory, providing different visual complexities and numbers of visual variables. To visually communicate the values of attributes, i.e., per-node or per-trajectory attributes, they can be mapped to visual variables such as color, line width, or the diameter of tubes or spheres. The different characteristics of the specific geometry types can further be utilized to make visual distinctions between trajectories, for example to depict different kinds of trajectories, to communicate interaction and exploration states, or to implement optimization methods such as level-of-detail techniques. In the following, the characteristics of the supported geometry types (Fig. 5.1) are explained:

Lines Trajectories are represented by thin lines, depicting the geo-positions of a moving aircraft. Color can be used to convey attribute values along the trajectory.



(a) *lines* (b) *tubes* (c) *ribbons* (d) *spheres*

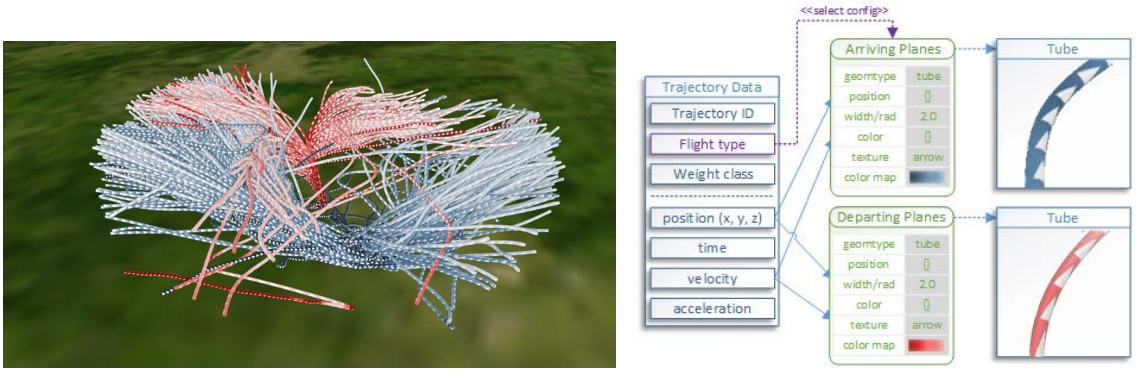
Figure 5.1: *Supported geometric primitives for direct trajectory visualization: lines (a), tubes (b), ribbons (c), and spheres (d).*

Tubes By extruding the geometry with a radius around the central line of movement, a 3D tube is created. In addition to its color, the tube's radius constitutes an additional visual variable for mapping attribute values. A texture map that is applied to the tube can either convey static information, such as the direction of movement, or it can provide further visual variables. The choice of texture itself provides one visual variable, another is obtained by modifying the stretch factor along the trajectory, which can express for example velocity. Finally, the texture map can be animated, using its speed as another visual variable.

Ribbons Instead of solid round tubes, lines can also be extruded to 2D ribbons within 3D space. Ribbons provide the same visual variables as tubes, but they can be harder to read due to their asymmetric shape. On the other hand, an additional degree of freedom arises from rotating the ribbons around their center, which can be used to convey additional information, for example the orientation of aircrafts.

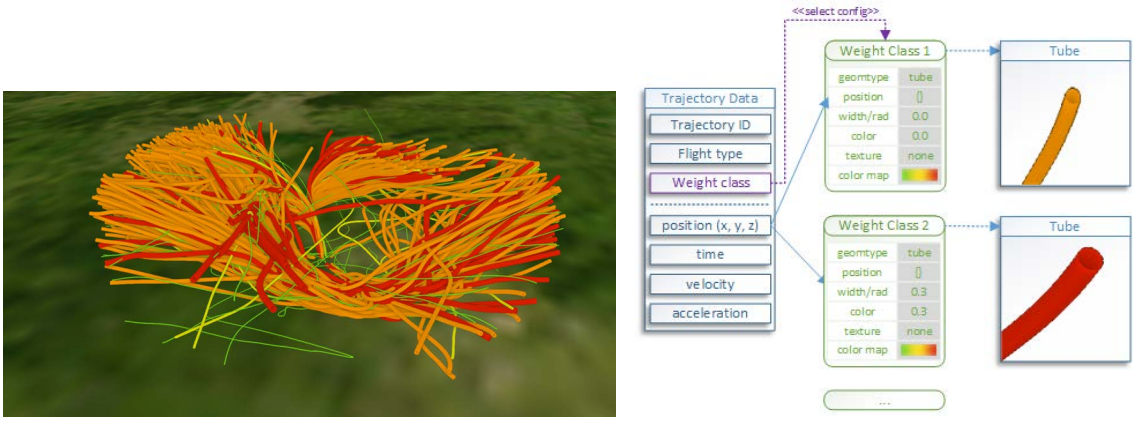
Spheres For detailed inspection, the values of per-node attributes can be of importance. These can be difficult to perceive with continuous depictions of trajectories such as lines or tubes. Therefore, to depict the attribute values of individual tracking points along a trajectory, nodes can be rendered as individual spheres instead, providing radius and color as two independent visual variables.

With respect to the use cases identified in Sec. 5.1, direct trajectory visualization can be applied to analyze single trajectories as well as to explore large trajectory data sets. The main tools for analysis are interactive mapping, exploration, and filtering. Mapping is applied to visualize the characteristics of trajectories with regard to their spatial and temporal features, and their attribute values (e.g., velocity or acceleration). Using this approach, analysts can visually find patterns of interest in the data set. Then, the field of search can be narrowed by exploration and filtering to continue the analysis in more detail, e.g., on smaller numbers or even individual trajectories. In the following, this is demonstrated using the data set described in Sec. 5.3.



(a) *Resulting visualization* (b) *Applied mapping configuration*

Figure 5.2: *Inspection of a data set by classification into departing and approaching aircrafts.*



(a) *Resulting visualization* (b) *Applied mapping configuration*

Figure 5.3: *Inspection of a trajectory data set by classification into four weight classes.*

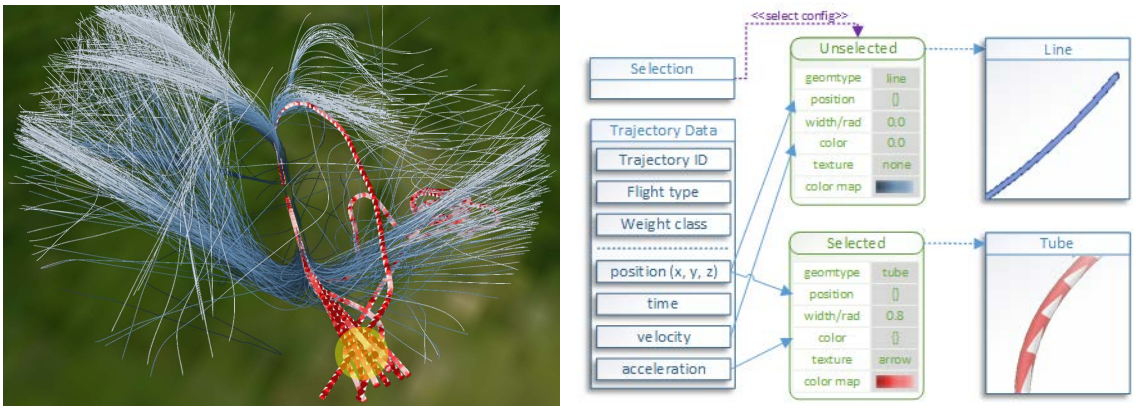
Overview Visualization To get an overview over the trajectories, dynamic mapping is applied to identify different classes of trajectories (A.3). In Fig. 5.2, approaching and departing aircrafts are visualized with two distinct color schemes, blue and red, respectively. Texture mapping and animation are used to communicate direction (arrow texture) and velocity (texture stretch and animation speed). In the resulting visualization, the number of approaches and departures can be roughly estimated. Also, common flight routes for approach and departure can be clearly identified, as well as outliers (A.4).

Another classification can be applied by visualizing the weight class of aircrafts. In Fig. 5.3, an appropriate mapping configuration is used that expresses the weight class of aircrafts by the thickness and color of their trajectories. The resulting image reveals mainly large aircraft types (red and orange) visiting the airport.

Spatial Exploration Instead of classifying trajectories by their attribute values, the spatial distribution and routes of individual flights can be of interest for trajectory analysis. For example, an analyst's objective might be to identify all flights that passed through a specific region, or - in combination with temporal filtering (see Sec. 5.6) - all flights that moved through a region in a specific time span. This can be achieved by using a visual lens, which selects all trajectories that intersect with it, and applying a mapping configuration that highlights the selected trajectories in the focus region.

Fig. 5.4 shows an example focus+conent mapping. An interactive lens is used to select trajectories in a certain region, which are displayed in more detail and using a different color scheme. Using this type of spatial exploration and filtering, a subset of trajectories can be selected interactively for further detailed analysis.

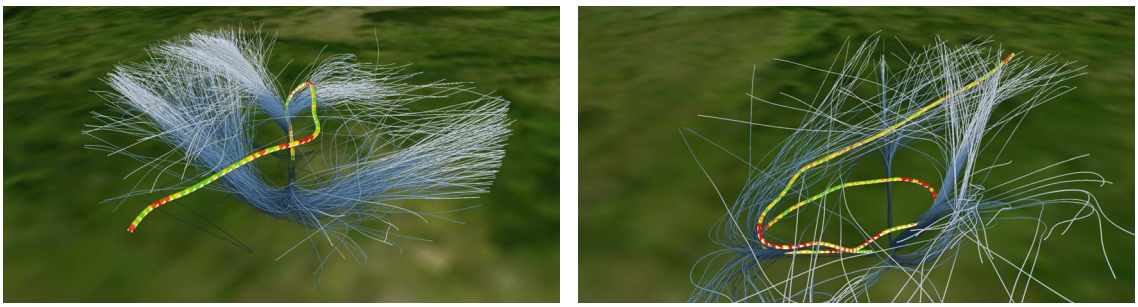
Detailed Inspection Individual trajectories can be identified and chosen for further analysis (A.1) by interactive filtering, selection, and highlighting. Fig. 5.5(a) shows the selection of a single trajectory. It uses a mapping configuration such that the selected trajectory is visually highlighted and displays more details (acceleration is mapped to color). The remainder of the trajectories is displayed smaller.



(a) *Resulting visualization* (b) *Applied mapping configuration*

Figure 5.4: *Spatial exploration example: An interactive lens (yellow overlay) is applied to select trajectories in a certain region. The selected trajectories (red tubes) are visually differentiated from the unselected trajectories (blue lines).*

When an interesting trajectory is found, the mapping configuration can be adjusted to display the selected trajectory in even more detail, if not hiding other trajectories altogether by filtering. In Fig. 5.5(b), a trajectory with an unusual flight path is inspected by visualizing velocity (texture stretch) and acceleration (color). It represents a missed approach on an airport. To focus on the selected trajectory, the remaining trajectories are reduced to line rendering.



(a) *Analysis of aircraft trajectories by interactive exploration and selection.* (b) *Detailed inspection of an individual trajectory representing a missed approach on an airport.*

Figure 5.5: *Detailed inspection of trajectories by selection and dynamic mapping.*

Helper Geometry When analyzing large data sets with direct trajectory rendering, perception problems can occur. Visual clutter can be reduced by interactive filtering and decreasing the number of trajectories visualized at the same time. Other problems are occlusion and perspective foreshortening, which arise from visualizing 3D scenes using 2D images. They can have undesired effects on the visualization and analysis of 3D trajectories. For example, heights are hard to perceive and prone to misinterpretation, and due to improper height perception, movements can be misinterpreted largely. Fig. 5.6

demonstrates how helper geometry can be used to mitigate these problems (C.5): a shadow-like representation of the movement on the ground can help identify the true movement of an aircraft. Height perception can be improved by using a fence visualization. This can also be extended by using the fence for visualizing more attributes, however, this will also introduce more visual clutter and may increase the difficulty of interpreting and understanding such visualizations.

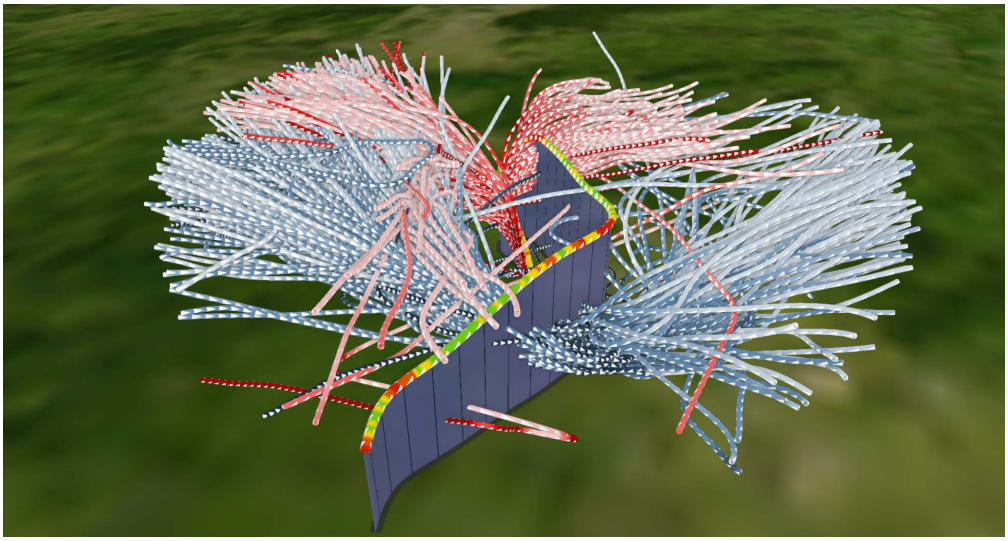
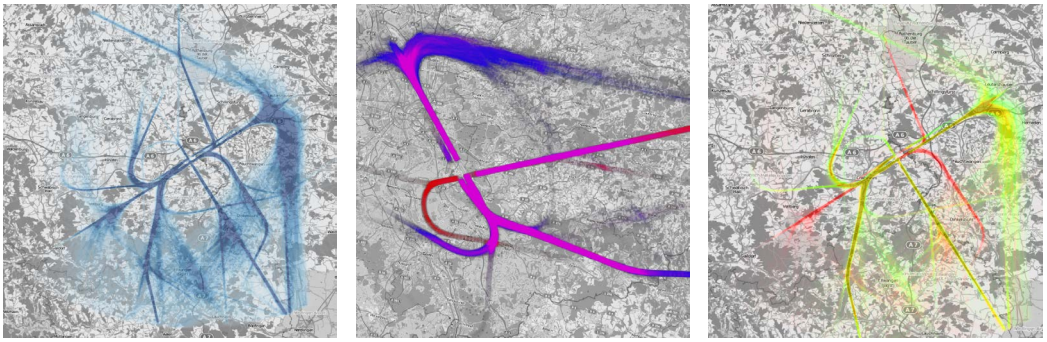


Figure 5.6: *Helper geometry generated in real-time to improve the perception of positions and heights in 3D trajectory visualizations.*

5.5 Interactive Density Maps

In addition to the visualization and analysis of individual attributed trajectories, movements can also be aggregated over time. Using a regular grid, this results in a density map that represents movement density during the selected time interval, i.e., the traffic volume at each spatial position on the grid. Density maps can either be precomputed using a static configuration or be created in real-time to reflect current filter options chosen by the user. By adjusting the temporal filter, users can interactively select the time interval they are interested in and get an instant update of the density map, reflecting the aggregated movements during the selected period. Further, attribute mapping options can be used to vary the influence of certain trajectories to the overall density, e.g., based on the speed or the weight class of an aircraft.

As a basic use case, density maps can be applied to assess the amount of traffic in a selected area and time interval. From this investigation, high density volumes can be exposed and common flight routes can be identified (A.2). Fig. 5.7(a) shows the resulting density map of the air traffic data set used before. It aggregates approximately 12,500 individual flights over a time interval of one month. The density map clearly shows common flight paths, corresponding to the runways of the airport and the routes for



(a) *Density map aggregating 12,5000 movement trajectories over the time of one month.*

(b) *Visual comparison of two density maps, implemented by blending*

(c) *Visual comparison of two density maps, implemented by difference visualization*

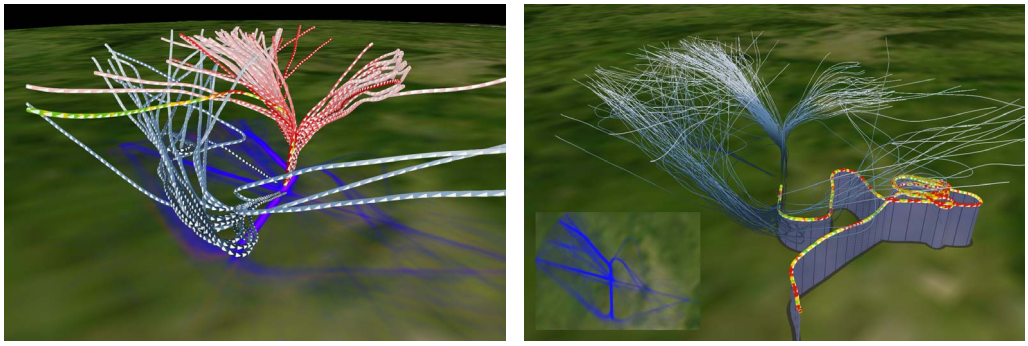
Figure 5.7: *Interactive density maps: overview visualization (a), blending of two density maps (b), and difference map visualization (c).*

arrival and departure as defined by air traffic regulations. Density maps can also be used to compare traffic volumes in different situations, such as specific time intervals or planning scenarios (A.6). By applying complement color schemes to two distinct density maps and blending the images together, disjunct areas and areas of intersection can be perceived (Fig. 5.7(b)). Another technique for comparing density maps is by calculating the numerical differences between traffic volumes, and mapping that difference to color using a diverging color map (Fig. 5.7(c)).

Finally, density maps can be applied for detecting outliers (A.4), i.e., aircrafts that do not follow common flight paths. By combining trajectory visualization and density maps, individual trajectories can be visually compared to regions of high density in the density map (Fig. 5.8). While a combination of the two visualization methods simplifies spatial perception and correlation of trajectories to the density map, it also increases visual clutter and occlusion (Fig. 5.8(a)). Another approach is to separate the two views using an overview+detail concept (Fig. 5.8(b)): The density map represents the overview, while the direct trajectory visualization reveals a detailed view into the scene. Interaction and visual feedback methods can be applied to intensify the correlation between the two scenes, for example by highlighting a trajectory in both views simultaneously.

5.6 Interactive Temporal Exploration

For both, the direct and aggregated visualization techniques, methods are required to convey and explore temporal information. Concepts such as the space-time cube are effective for exploring movements in 2D space, since they use the third dimension as a temporal axis. However, this approach is not applicable when the 3D component is important for the current analysis task. Therefore, indirect methods such as animation and interaction have to be applied. An interactive approach for temporal exploration



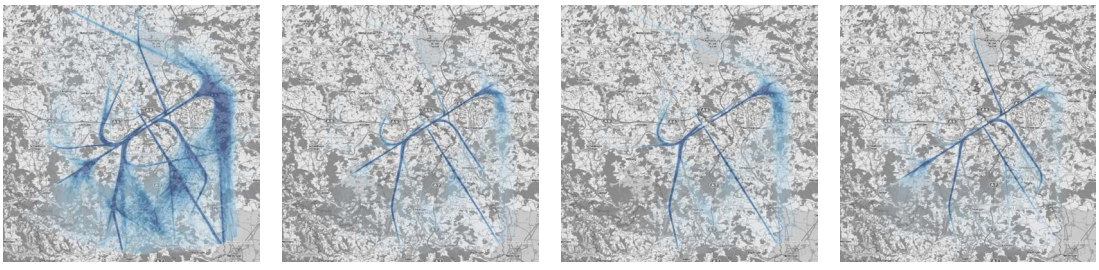
(a) *combined view* (b) *overview+detail*

Figure 5.8: *Combination of density maps with attribute trajectory visualization, integrated into a single scene (a) or as overview+detail views (b).*

seems promising, since it enables users to directly inspect and perceive temporal aspects of the movements based on the visualization techniques described before.

The easiest form of temporal exploration is implemented by enabling users to interactively filter input data by temporal aspects. A time slider control is used to interactively select and modify the inspected time interval. All active visualization methods, such as trajectory visualization or density maps, will be updated based on the subset of the data that lies within the specified interval. By continuously adapting and refining the time interval, a user can reveal temporal aspects of the data, such as hotspots, and find temporal correlations in the inspected data.

Fig. 5.9 shows the exploration of a density map from trajectories of one month. Originally, all trajectories are evaluated, producing a density map that reveals which flight paths have been used during that month. After that, the time interval is reduced to a week and moved dynamically over the month to discover changes in flight paths. As the visualization shows, certain routes are used more often in one week and nearly not at all in other weeks, while some flight routes are constantly used. By further refining and comparing, it can be investigated which flight paths are used under what circumstances.



(a) *Full month.* (b) *1st week.* (c) *2nd week.* (d) *3rd week.*

Figure 5.9: *Temporal exploration of trajectories from one month with an interactive 2D density map, refined by adjusting the temporal focus area.*

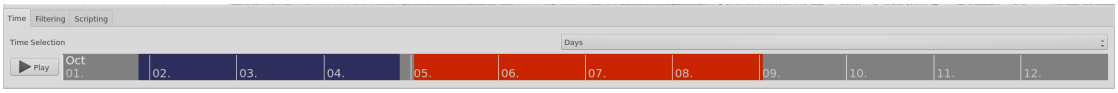
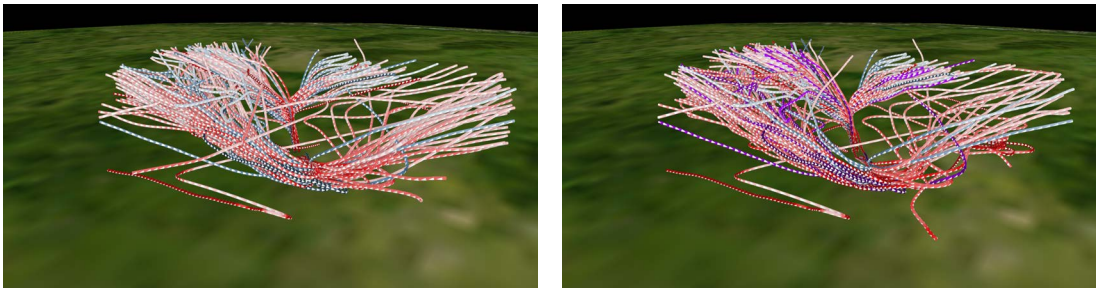


Figure 5.10: *User interface control for interactive temporal exploration. Two focus areas can be defined by adjusting the separate time spans (red and blue).*

Fig. 5.10 shows the user interface control for time interval selection used in the framework. It consists of a timeline displayed in the background and one or more highlighted areas in the foreground that indicate the selected intervals. Start and end time can be modified by dragging either side of the selection area, thereby extending or shrinking the selected interval. By panning the area, the time interval can be moved without changing its length. In analogy to spatial maps, the time interval can be zoomed in and out, changing its granularity (i.e., months, weeks, days, hours or minutes).

5.7 Temporal Focus+Context

The described exploration method can be used as a basis for a focus+context model [42] for temporal data. For that, all trajectories within a user-defined time interval are considered to be within the temporal focus, while the remaining trajectories represent the context. Depending on this information, the visualization techniques are parameterized to display focus and context differently. For example, trajectories in the context may not be disregarded entirely, but rather be visualized in a less prominent style. As a metaphor, the focus can be regarded as the foreground in a visualization, the context acts as its background: While data in the focus should be easily readable and visible in more detail, the context can be visualized in less detail. In order to visually compare different time spans, multiple temporal foci can be defined at the same time, using one time slider for each temporal focus. This allows a user to visually differentiate between the focus areas, the context, and overlapping areas between the different foci.



(a) *Two disjunct temporal foci are displayed by using different color schemes (red and blue).*

(b) *Two temporal foci are displayed, using a third color scheme (violet) to highlight the overlapping area.*

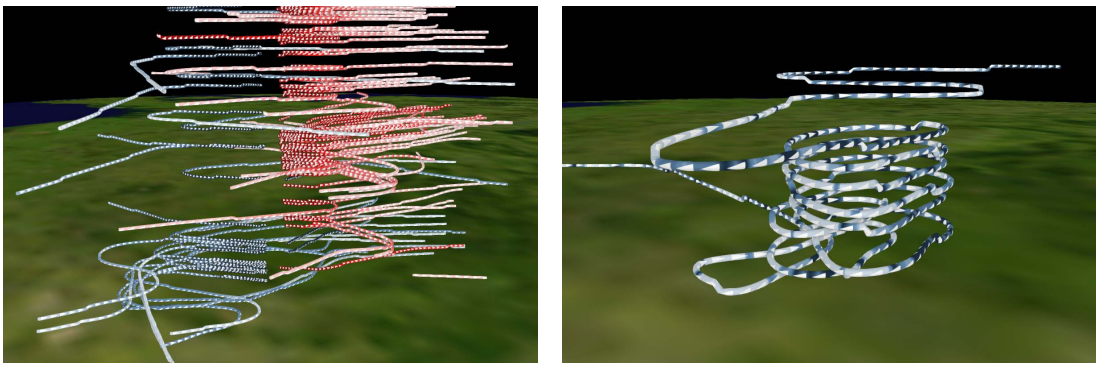
Figure 5.11: *Focus and context visualization using direct trajectory visualization. Several distinct or overlapping temporal focus areas are defined and can be visually identified by applying different attribute mapping configurations.*

For attributed trajectory visualization, interactive mapping is used to implement this temporal focus+context concept. A different mapping configuration is defined for each focus, the context, and the overlapping areas. Each of these can have a distinct visual style. This enables users to visually identify and compare the trajectories in different time intervals, e.g., to estimate the variation in numbers of trajectories. In Fig. 5.11(a), two such focus regions are defined at disjunctive time intervals. Trajectories within the selected time spans are depicted in red and blue respectively, to compare the number of trajectories and flight routes between to different time periods. Fig. 5.11(b) shows a similar setting and additionally visualizes trajectories within the overlapping time interval using a third color (violet). In the case of density maps, the context is usually omitted. Different focus areas are displayed using distinct color schemes.

5.8 Space-Time Cube

To express temporal relationships between trajectories (R.2), e.g., to compare the movements of a number of trajectories, to find intersections, or to understand the temporal order of flights (A.1), space-time cube visualization can be applied. Using dynamic attribute mapping, a configuration can be created that maps the time attribute to height. The height is determined in a way that the beginning of the selected time period is mapped to the ground level of the map, while later points in time linearly increase in height. This enables temporal exploration with the STC by adjusting the time slider.

In addition to a STC configuration for all trajectories, mapping can also be adjusted in a way that STC mapping is used only for selected trajectories. To get an overview over all flights in the region, the actual flight paths are displayed first. On selection of one or multiple trajectories, the mapping is switched to STC, which allows for their movement paths to be compared temporally. An example of a mapping configuration is displayed in Fig. 5.12: While Fig. 5.12(a) applies the space-time cube to visualize the temporal order of departing and landing aircrafts, Fig. 5.12(b) displays the movement of a single trajectory in its temporal detail.



(a) *Temporal order of departing (red) and arriving (blue) aircrafts*

(b) *Detailed visualization of the movement of a single trajectory over time, mapping acceleration to color*

Figure 5.12: *Interactive space-time cube visualization.*

Chapter 6

Climate Network Analytics

Climate network analysis is a recent method in the field of climate research that applies complex network theory onto climatological data [187, 207, 53]. With this method, the dynamics of the climate system are represented by a network structure, which expresses statistical interrelationships between different regions. Climate networks are constructed from climatological data by applying a measure of correlation or dynamical similarity between regions, for example similarities of time series, which are represented by edges in the network. Thus, a network of nodes and edges representing the climate data is constructed:

> "The vertices of a climate network are identified with the spatial grid points of an underlying global climate data set. Edges are added between pairs of vertices depending on the degree of statistical interdependence between the corresponding pairs of time series taken from the climate data set" [52].

The analysis of climatological data using climate networks is conducted by applying methods of complex network theory to the constructed networks. Thus, network measures such as degree or betweenness centrality are derived from the network's topology, in order to analyze the structure of the network and to investigate the spatio-temporal variability of the network. This chapter presents an approach for interactive visualization and analysis of climate networks using a visual analytics toolset.

6.1 Use Case

The main objective of climate network analysis is to derive insights into the climate system by analyzing the structure of the constructed network. To this end, statistical methods mainly based on network measures are applied. However, an interactive visual analytics approach can be helpful for the visualization and exploration of network features, for visual pattern recognition, and finally for the communication of results. Based on a survey conducted with climate researchers, the following tasks and challenges have been identified for visual analytics of climate networks:

A.1 Analyzing Network Structure In general, researchers stated they were interested in "getting familiar with the network's structure". This involves a clear representation of edges, which represent the structure of the network. Also, network measures especially on the edges need to be communicated.

A.2 Finding Unknown Patterns To find patterns in the network data, the analysis is based mainly on network measures, which are represented as attributes on nodes and edges. Therefore, the network measures need to be clearly communicated. Also, filtering the network by network measures helps finding patterns by emphasizing selected structures and focusing on subdivisions of the entire network.

A.3 Identifying Clusters Another objective lies in finding clusters (or communities) in the network, i.e., groups of nodes and edges that form strong connections between parts of the networks. Therefore, tools for finding groups of nodes and edges are needed. In addition, filtering is needed to focus on the clusters and emphasize the identified structures.

A.4 Identifying Hierarchical Structures In addition to groups and clusters, another task is to reveal implicit hierarchies in the network structure. For example, local patterns may be supported by larger backbone structures in the network. To find such hierarchies, tools for multi-scale analysis and filtering are required.

A.5 Linking Network Structure to Geographical Context Due to the geophysical nature of the climate data, a strong connection of the network data that is being analyzed to the geographical context in which the original phenomena occurred exists. This connection can be important for analyzing and understanding climate networks, for example to link the structures and phenomena identified by network analysis to the geographical background such as geographical features (landscape) or land use. This can be achieved by connecting the network data with additional geographical data, represented for example by geographic or topographic maps.

6.2 Data Characteristics

A climate network is represented by an attributed, undirected, geo-referenced graph $G = (V, E)$. V denotes a set of nodes, containing the geo-position of the respective node and data attributes that include additional information about the geographical location, e.g., temperature or precipitation. E denotes the set of edges, containing the pair of nodes connected by the edge and edge attributes, that represent data about the relation.

To adapt the developed visual analytics system for climate network analysis, several types of data sets have to be taken into account. They vary not only in data size, but also for example in the scale of analysis (global, regional, or local networks) and data characteristics (e.g., 2D networks, or 3D coupled networks). In the following, the categories and characteristics of data sets that have been used in this work are described, identifying the challenges for developing interactive visualization and exploration tools for the analysis of climate networks.

Large Graphs Climate research is conducted on different scales, namely local, regional, and global. Depending on the scale and origin of the data, the resulting climate networks vary in size from 1.000 to over 300.000 nodes, often with a high edge density. The number of edges range from a few thousand to millions of edges.

Visualization methods for climate networks therefore need to be able to process large numbers of nodes and edges in real-time while maintaining interactivity. The large numbers of nodes and edges also pose visual challenges such as edge clutter, which need to be addressed. Also, the density and distribution of nodes varies largely depending on the data source. For example, satellite data usually results in a regular grid of nodes, in contrast to station based data. Visualization methods need to be adaptive to different scales, ranging from local (e.g., a city or county) to global (the entire earth).

Geo-Referenced Data The nodes in a climate network are geo-referenced, representing a specific position on earth. Edges, on the other hand, although representing a connection between two geo-referenced nodes, do not carry a geographic meaning themselves: they merely describe a statistical relationship between nodes. As a consequence, while node positions are fixed, the routes of the edges as chosen by the visualization technique do not necessarily correspond to geographic phenomena (e.g., the direction of currents).

This characteristic is important for the visualization of climate networks. On the one hand, edges should be visualized in a way that connections between two nodes can be easily identified. This is usually accomplished best by a straight line between the nodes. On the other hand, a visualization should try to avoid the misinterpretation of edges having a geographic significance. Further, when embedded into a geo-virtual environment, connections can easily be misinterpreted in length. For example, when using 2D map projections, connections between nodes on different sides of the projection may seem very long, although having a short geo-distance in reality.

Attributed Data Climate network data is usually attributed, carrying additional data attributes on both nodes and edges. Those attributes originate either directly from the source data, e.g., temperature or precipitation measures, or are derived network measures, such as degree, density, or betweenness. On average, climate networks contain 5 to 15 additional attributes on nodes and edges. The challenge for visualization lies in communicating these attributes together with the network structure itself. Attribute values need to be clearly readable and explorable, while still maintaining the visualization of the overall network structure expressed by its nodes and edges.

Coupled Networks Some climate networks, called *coupled networks*, contain three-dimensional data. They carry a height value in addition to the geo-position of nodes, representing for example elevation or atmospheric levels. The nodes and edges on each level constitute separate partitions, which may be analyzed separately. In addition to that, interconnections between different partitions can be of high significance for the analysis and therefore constitute another partition of their own.

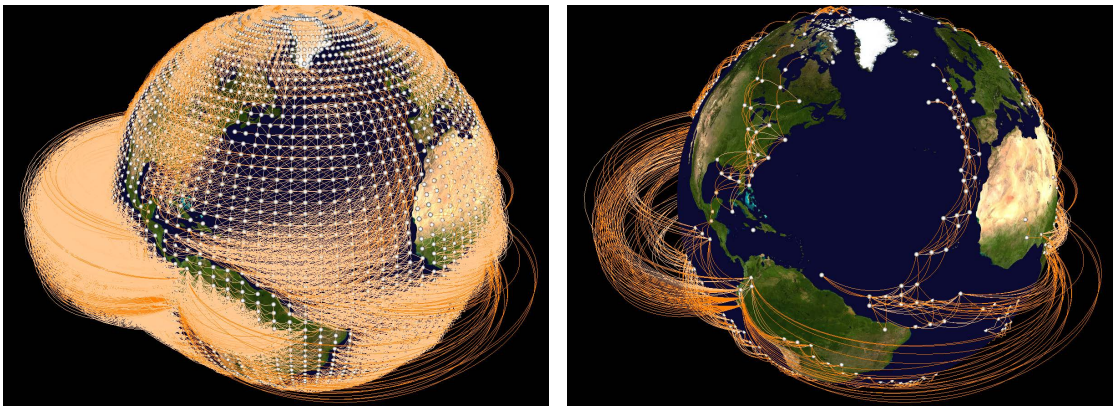
From a visualization perspective, the 3D positions of nodes restrict the use of the 3rd dimension for other purposes, e.g., for the visualization of edges using 3D arcs. Also, the coupled nature of the network needs to be visualized separately to emphasize on the distinct partitions and inter-level connections. For example, different color maps and visual styles should be applied to visualize different parts of the network and the interconnections between the parts separately.

Temporal Networks Climate networks can also contain a temporal dimension. For example, *evolving networks* may represent the evolution of a network over a number of years. Temporal filtering and selection tools need to be applied to enable the interactive exploration of the changes in a network over time. Also, time-dependent networks often significantly increase the amount of data that needs to be handled, as the number of nodes and edges are multiplied by the number of time points represented in the data set.

6.3 Geo-Referenced Network Visualization

Using the GPU-based visualization pipeline described in this work, an interactive visualization method for climate networks has been implemented. Climate networks are represented as node-link diagrams, embedded into a geographical map (2D) or globe (3D) for cross-referencing network data with topological or thematic features (R.1). The nodes of the network are depicted by spheres, which are rendered using splats [155] to improve rendering performance (C.2). This enables large amounts of nodes to be rendered in real-time with low memory consumption for the generated geometry (C.1). The spheres provide radius and color as visual variables onto which node attributes can be mapped. Edges are represented by lines between the connected nodes, which are rendered either as 2D lines or 3D arcs. The line color, line width, and arc height are available as visual variables for the mapping of edge attributes.

Due to the embedding of climate networks into the geographic background, node positions need to be fixed according to their geo-position, thus position-changing layouting of nodes to improve the perception of the network structure cannot be applied in this context. Therefore, edge clutter produced by long and overlapping edges can occur, especially in highly connected and global climate networks. To reduce such problems, edges can be rendered as 3D arcs instead of 2D lines. This reduces edge clutter, since



(a) *Unfiltered view of a global climate network with 6000 nodes and 115 000 edges.*

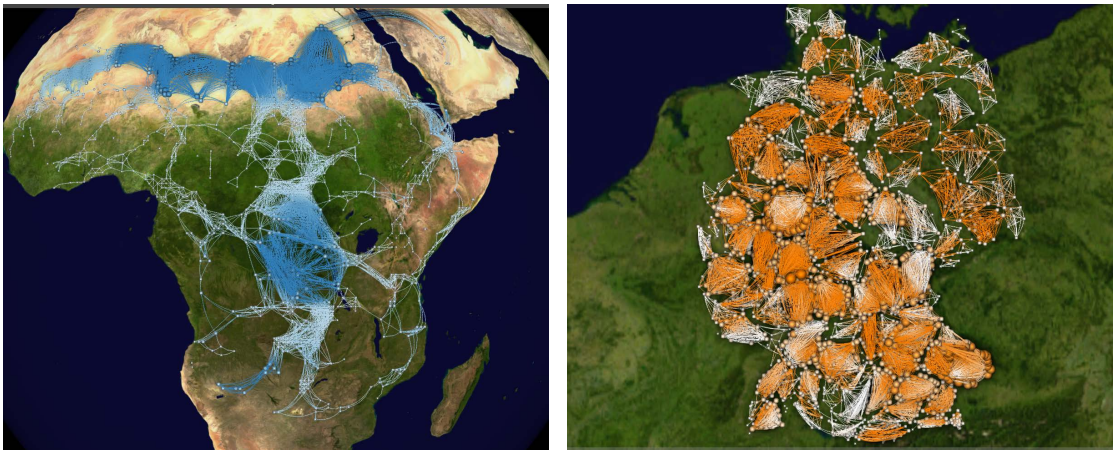
(b) *Filtered view on global climate network to highlight edges with large shortest-path betweenness.*

Figure 6.1: *Visualization of a global surface air temperature network, displaying the entire unfiltered network (a), and filtered by edge-betweenness (b).*

edges overlap more rarely in the 2D plane. It can also improve the perception of long and short distance edges by varying the height of arcs with their length, which is important for understanding the main network structure (A.1). On the other hand, a 3D visualization can also impede the perception of the visualized data. Therefore, additional techniques such as filtering and highlighting are applied to improve the visual perception of the network structure.

Figure 6.1(a) shows a visualization of a global surface air temperature climate network [52]. It contains approximately 6000 nodes and 115 000 edges before filtering, which are displayed in real-time. The resulting visualization shows the grid points of the underlying climate model as nodes of the network. Short- and long-range edges can be clearly perceived by applying 3D arc rendering. The color of nodes and edges represent node or edge betweenness, respectively. In Figure 6.1(b), interactive filtering has been used to filter the network by edge-betweenness, showing only edges with large shortest-path betweenness. The climate researcher thereby highlights structures of matter and energy flow in the climate system that are known to follow the shortest paths in the networks [136, 188].

In Figure 6.2, two examples for smaller climate networks (regional and local) are displayed. Figure 6.2(a) shows the visualization of a rainfall network of Africa. It is filtered to display nodes with high shortest-path betweenness, thus highlighting the regions that correspond to moisture convergence zones. In addition to that, node degree is mapped to node color and size, and betweenness is displayed by edge color to convey the structure of the network. In Figure 6.2(b), a local temperature network of Germany is displayed that represents data from local weather stations. The network is unfiltered, yet the localized structure of the network is clearly visible and emphasized by displaying node degree in both color and size, and betweenness in edge color.



(a) *Rainfall network of Africa, displaying node degree in size and color, and betweenness in edge color.*

(b) *Temperature network of Germany, displaying node degree in size and color, and betweenness in edge color.*

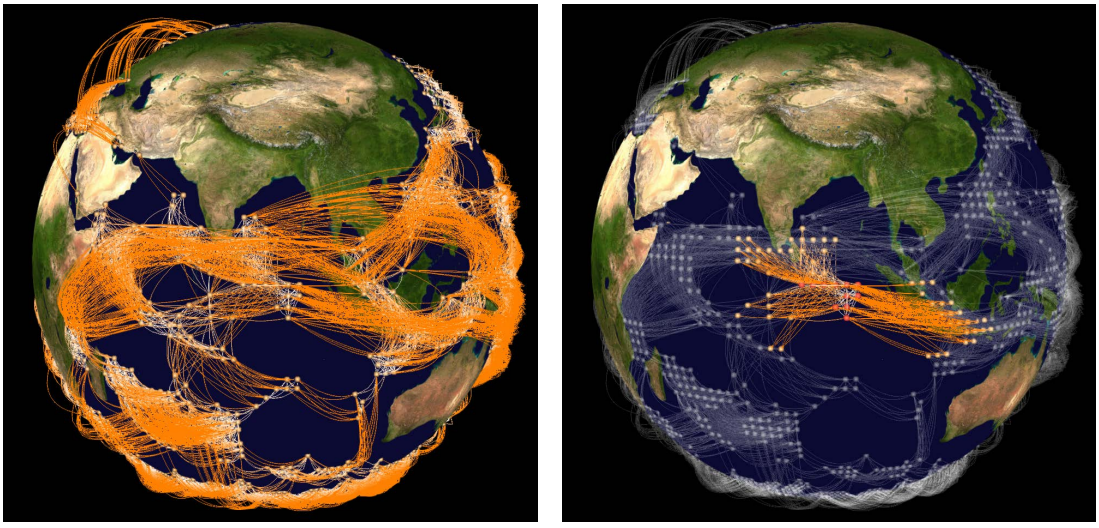
Figure 6.2: *Visualization of regional (a) and local (b) climate networks.*

6.4 Interactive Filtering and Selection

As a tool for understanding network structure, filtering is applied. It narrows the visualization to certain parts of interest, thereby revealing structures and correlations in the data (A.1). Further, it reduces visual overload and clutter by reducing the number of nodes and edges that are displayed simultaneously. For example, global networks can be analyzed by reducing the number of nodes and edges to a size that can still be visualized interactively, so only global and long-range patterns are displayed (e.g., by filtering short edges). Then, sub-networks can be selected for further detailed analysis by filtering the long edges and focusing on smaller structures.

Interactive filtering is also important for formulating and testing hypotheses on the underlying climatic processes (A.4). By interactively changing the filtering options, correlations between attributes such as shortest-path betweenness can be tested and verified. This has been demonstrated in Figure 6.1: By filtering the edges by node-betweenness, the structure of the network with regard to energy flow could be revealed.

Another tool for analyzing network structure is interactive selection. In Figure 6.3(a), a global sea surface temperature network is visualized. The structures of dependency between grid points have already been revealed by filtering the network to remove nodes with low betweenness centrality. However, the number of edges and the amount of overlapping edges is still very high. To better analyze the connections between specific regions, nodes can be interactively selected. The selected nodes and connected edges in a k-neighborhood, whose size can also be adapted interactively, are visually highlighted. The remainder of the network is visualized with high transparency, but not filtered out completely, as shown in Figure 6.3(b). Thus, the structure of a network can be analyzed in an interactive process, by progressively refining the regions of interest.



(a) *View of the network filtered by betweenness-centrality, without selection.*

(b) *View of the network with selection to highlight the connections in a specific region.*

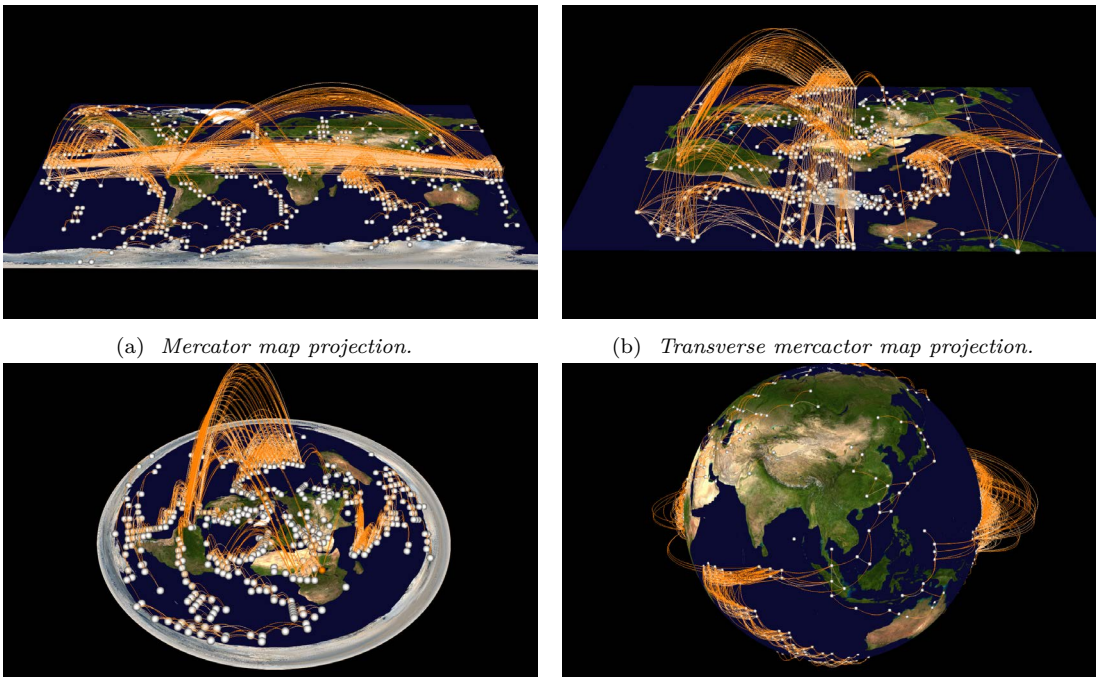
Figure 6.3: *Sea surface temperature network without (a) and with interactive selection (b).*

6.5 Dynamic Map Projections

By embedding networks into a geo-virtual environment, researchers can correlate network data with geographical features. To this end, either 2D map projections or spherical representations such as 3D virtual globes can be used. The virtual environment however can have a big impact on the perception of networks in their geographic surroundings.

Choosing 2D map projections can result in a distortion of neighborhoods, as most common map projections are not distance preserving. Therefore, the perceived distance on the map does not always represent the actual geo-distance. Map projections can also increase visual clutter due to large overdraw of edges. Due to the projection, nodes can end up at opposite sides of the map, although they are rather close together in reality. Edges between them would cross the entire map and thereby give a wrong impression of their actual distance and clutter the image. Splitting up edges at the opposite borders reduces edge clutter, but disturbs the perception of connections between nodes.

With spherical representations in the form of a 3D globe, some of these problems can be reduced. Geo-distances are better preserved. On the other hand, a 3D globe representation results in occlusion of at least half of the network, which is always hidden on the backside. Also, 3D visualization is often critiqued for hindering perception of geographic information and 3D interaction methods are often perceived to be harder to handle. Therefore, the choice between 2D and 3D visualization is often difficult [54].



(a) *Mercator map projection.* (b) *Transverse mercactor map projection.*

(c) *360 degrees map projection.* (d) *3D virtual globe.*

Figure 6.4: *Supported dynamic map projections for global networks: Mercator (a), Transverse Mercator (b), 360 Degrees (c), and 3D Globe (d).*

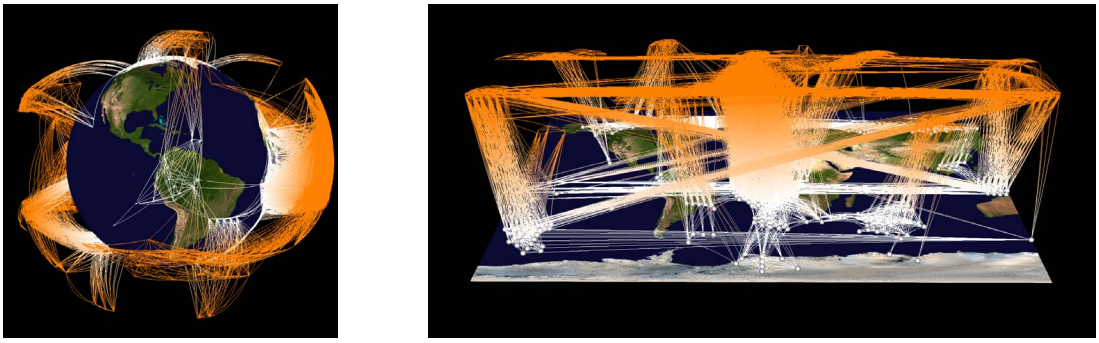
To enable researchers to choose the appropriate visualization of their data according to their current research task, the visualization framework supports several 2D map projections as well as a 3D virtual globe as a background for geo-data visualization. The geographic background can be switched interactively between the map projections and between 2D and 3D representations. Since the geographic projections have been implemented in shader programs, which are evaluated as part of the rendering process, no recomputations or reconstructions of geometry is necessary, so the transition between different geographic backgrounds is immediate.

Figure 6.4 shows the differences between several types of projections in visualizing the same global climate network. While a standard Mercator projection (Figure 6.4(a)) exhibits strong edge clutter around the equator region, this effect is avoided by the transverse Mercator (Figure 6.4(b)) and spherical (Figure 6.4(c)) projections. This also applies to the 3D representation (Figure 6.4(d)), but here only one hemisphere is visible.

6.6 Coupled Network Visualization

Coupled climate networks consist of several interconnected layers of nodes, which represent for example different atmospheric levels. To visually analyze such coupled networks, a strong visual distinction between each layer and additionally the cross-layer connections is important. This can be achieved for example by assigning different color schemes to different layers, or by mapping height to color using a single color scheme.

Figure 6.5 shows a visualization of a coupled climate network, constructed from the geopotential height field on two isobaric surfaces [51]. The network has been filtered to highlight nodes with large cross-betweenness, indicating regions that are potentially important for vertical interactions in the atmosphere. Color mapping has been applied to highlight the two layers as well as cross-layer edges by mapping height to color.



(a) *3D coupled network with a geocentric view.*

(b) *3D coupled network with a Mercator map projection.*

Figure 6.5: *Visualization of a coupled 3D climate network using two different geographic projections: 3D geocentric view (a), and 2D Mercator map projection (b).*

Chapter 7

Conclusions

In this chapter, the presented concept and implementation of a GPU-based visualization framework is evaluated and discussed. First, a performance evaluation for the proposed visualization method for massive 3D trajectories is performed with regard to rendering and update performance. For comparison, the implementation is tested against four alternative implementations. In the second part, the concept and its application for visual analytics of geo-referenced data are reviewed and discussed.

7.1 Performance Evaluation

The concept of a GPU-based visualization framework, as described in Chapter 3, is characterized by uploading data sets onto the graphics card in the form of attributed vertex clouds (Chapter 3.3.2) and using shader programs to process, transform, and finally visualize the data sets solely on the GPU. In contrast to traditional GPU-accelerated visualization methods, which perform the preprocessing, filtering, and mapping tasks mostly on the CPU, generating a geometric representation which is then rendered efficiently on the GPU, this method especially allows for complex filtering and mapping operations to be executed interactively, as costly regeneration of complex geometries and memory transfers between CPU and GPU memory are avoided.

To assess the performance of this approach and compare it to other implementations, two performance measures have to be taken into account: *rendering performance* and *update performance*. Rendering performance is measured by the time it takes to render a single frame for the visualization of a given data set. It includes the total sum of all stages of the programmable rendering pipeline, i.e., geometry processing and rasterization. Update performance describes the amount of time it takes to update the parameters for the visualization, for example by choosing different mapping or filtering options for the visualization. Depending on the implementation, this may include the time to regenerate geometric representations for the displayed portion of data.

For performance comparison, five alternative implementations of the visualization method for attributed 3D trajectories have been developed. They are functionally equivalent and produce the same visual result, but use different paradigms to generate and render the 3D trajectory geometry. Two GPU-based implementations follow the concept of the GPU-based visualization pipeline, using only different types of shader implementations to generate the actual geometry. In contrast to that, three different

CPU-based implementation have been implemented, which apply a typical rendering approach: they generate the geometry for the visualization on the CPU and upload it to the GPU for rendering.

Indexed Triangles (CPU) In this implementation, the geometry for the entire data set is stored in a single vertex array. It is referenced by an index array, which represents the list of individual triangles that form the 3D tubes. Therefore, the tubes for the entire data set can be rendered with a single draw call. However, the memory consumption of this approach is relatively large, since vertices that are shared between adjacent triangles must be repeated in the index array. Triangle strips cannot be used in this approach, as that would wrongly connect the different trajectories to each other, which are stored sequentially in the index array.

Triangle Strips with Primitive-Restart (CPU) This implementation also stores the geometry of the tubes using one vertex array and one index array, but instead of triangles, triangle strips are stored. This significantly reduces the number of indices needed to describe a single tube. To separate the individual trajectories from each other, an OpenGL feature called *glPrimitiveRestart* is applied: this instructs the rendering pipeline to start a new triangle strip each time a certain index value is encountered. This allows us to render several independent tubes within a single draw call.

Triangle Strips with Multiple Draw Calls (CPU) As an alternative solution to separate the independent trajectories from each other, this implementation uses different draw calls for each individual trajectory. The geometry is stored as triangle strips in a single vertex buffer. No index buffer is needed in this case, which improves the rendering performance as no lookups are required during rendering, however, multiple render calls can have a huge impact on rendering performance.

Geometry Shaders (GPU) This implementation applies the concept of the attributed vertex cloud to represent complex input data on the GPU. Data is stored in a vertex array and processed by a single draw call. Geometric representations for the data are dynamically generated during the rendering process, using vertex and geometry shaders, so no updates to the data are necessary when visualization parameters have been modified.

Tessellation Shaders (GPU) This implementation is based on the same attributed vertex clouds, but uses the more recent concept of tessellation shaders to generate the geometry. This approach offers less flexibility with regard to generating arbitrarily complex geometries, but in return provides a better rendering performance.

For the performance evaluation, the trajectory data set as described in Section 5.3 has been used. With each of the five implementations, rendering performance for a single frame has been measured, visualizing a subset with either 1000, 10.000, or 100.000 trajectories. Then, update performance has been measured by alternating between two sets of mapping parameters, which require the geometric representations to be regenerated (switching between tubes and spheres geometry). All performance tests were repeated 1000 times, and the average execution time has been determined. The performance evaluation has been conducted with a consumer graphics card (GeForce GTX 960M).

Table 7.1 shows the results of the rendering performance analysis (see also Figure 7.1). The three CPU-based visualization pipeline implementations perform equally well at about 8-9 ms per frame for 100.000 trajectories. In comparison, the GPU-based visualization pipeline implementations perform worse at a factor of 3. This result is not surprising, as the CPU-based visualization pipelines use the GPU for the rendering of a static geometry, which has been generated once on the CPU. Therefore, the rendering pipeline has to perform only geometry processing and rasterization and can therefore be executed very fast. The GPU-based implementations on the other hand generate new geometry as part of the rendering pipeline in each frame, which explains the slower execution. As expected, the implementation based on tessellation shaders performs about 25% faster than the implementation with geometry shaders.

	1000	**10 000**	**100 000**
gpu-geometry	337831 ns	2975580 ns	29514223 ns
gpu-tessellation	248592 ns	2138724 ns	21318387 ns
cpu-strips-multiple-calls	90747 ns	1496613 ns	8863904 ns
cpu-strips-primitive-restart	91937 ns	794214 ns	7859794 ns
cpu-triangles-indexed	92384 ns	800180 ns	7974709 ns

Table 7.1: *Comparison of rendering performance for five different implementations: time in nanoseconds to render a single frame for either 1000, 10 000, or 100 000 vertices.*

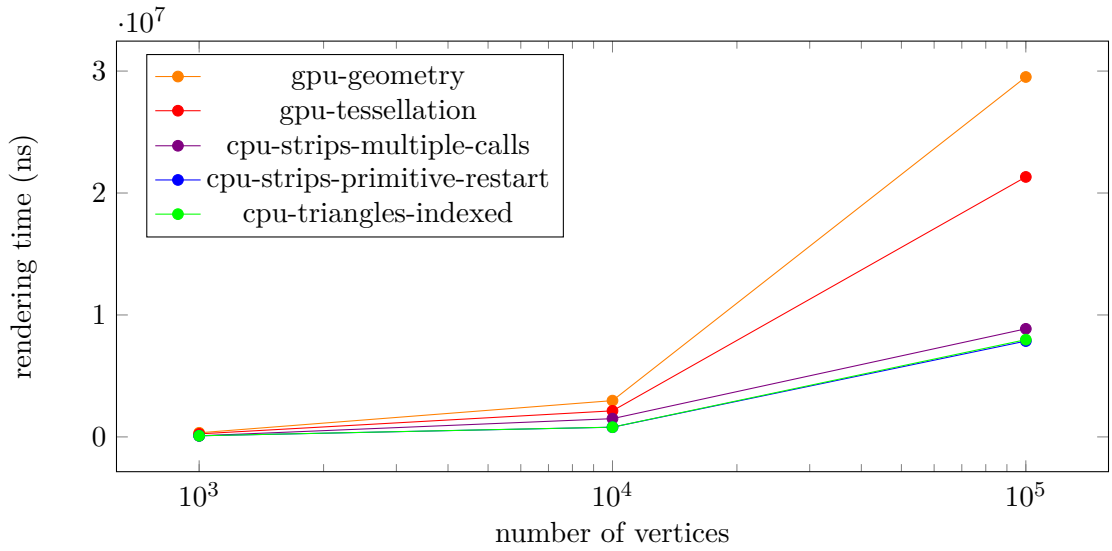


Figure 7.1: *Comparison of rendering performance for two GPU-based implementations (orange and red), and three CPU-based implementations (violet, blue, and green).*

Table 7.2 shows the results of the update performance analysis (see also Figure 7.2). For the CPU-based implementations, an update to the mapping configuration for 100.000 trajectories requires approx. 0.7 seconds. This result is not surprising, as the geometry has to be recreated and uploaded to the GPU. The GPU-based implementations on the other hand take only a few micro seconds for the update, since the mapping configurations require merely a modification of uniform shader variables. A high update time for CPU-based implementations can hardly be prevented, as modifications of filtering and mapping options may effect any number of elements in the data set, so with the desired flexibility and interactivity in mind, it is hard to optimize. This will be discussed in further detail in the next section.

	1000	**10 000**	**100 000**
gpu-geometry	83592 ns	76310 ns	78206 ns
gpu-tessellation	85222 ns	78267 ns	88164 ns
cpu-strips-multiple-calls	4172299 ns	44712203 ns	695020369 ns
cpu-strips-primitive-restart	4609966 ns	50453185 ns	716748647 ns
cpu-triangles-indexed	4262235 ns	50487066 ns	778268694 ns

Table 7.2: *Comparison of update performance for five different implementations: time in nanoseconds to update the geometry for a data set with either 1000, 10 000, or 100 000 vertices.*

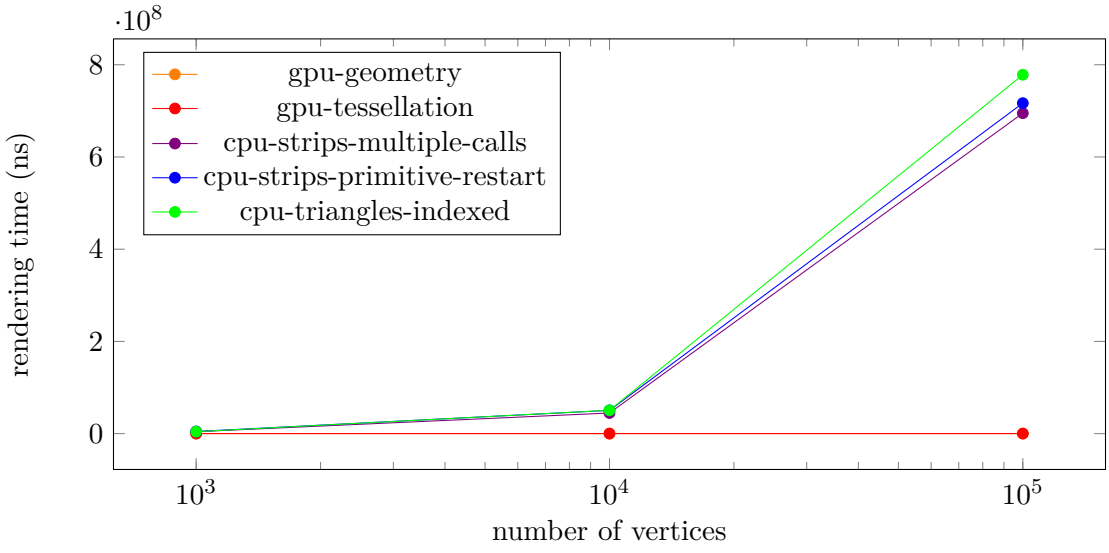


Figure 7.2: *Comparison of update performance for two GPU-based implementations (orange and red), and three CPU-based implementations (violet, blue, and green).*

7.2 Discussion

This thesis explored the concept of applying the GPU in the visualization process for more than efficient rendering of large geometries. Instead, all significant stages of the visualization process, including preprocessing, filtering, mapping, and geometry generation, are executed on the GPU during the rendering process.

The approach is based on the assumption that in visual analytics settings, large data sets are often processed and interpreted by human analysts in an interactive process. In order to explore the data and make sense of it, finding structures and testing hypotheses, they need to apply interactive tools such as filtering, mapping, and selection. This helps the analysts to interpret previously unknown data in an interactive way.

To enable this kind of interactive analysis, the complex input data that is to be visualized is represented directly on the GPU, using the concept of an attributed vertex cloud. Instead of generating geometric representations for the data once and uploading them to the GPU for rendering, they are dynamically generated during the rendering process. This enables a modification of visualization parameters at nearly no update costs, so the results of mapping, filtering, or selection actions made by the analyst are immediately visible. Also, additional analytical representations of the data, such as density maps, can be derived at the same time during the rendering process, taking into account the current filtering and mapping parameters.

The concept has been demonstrated for two diverse use cases. The first, interactive rendering of 3D movement trajectories, exemplifies a visual analytics system for a specific class of spatio-temporal data that originates from sensory data, as generated by air-traffic control systems around the world every day. In this case, the concept has been applied to demonstrate an interactive tool for the exploration of historic data sets. It includes the real-time visualization of 3D trajectories, the management and processing of temporal data sets on the GPU, and real-time generation of derived data such as density maps, which are included into the interactive visualization process.

The second use case, visualization of large geo-referenced climate networks, is an example of publicly available data, in this case climate data, which requires specialized tools for the interactive analysis and communication of analysis results. The concept was in this case applied to enable the visualization of much larger data sets as the currently available graph visualization frameworks are able to process. Due to the flexibility of the GPU-based visualization pipeline, spatial, temporal, and attribute based filtering and mapping operations have also been implemented and integrated into the interactive visualization process. Also, it enabled the implementation of specific requirements for the use case, such as the interactive switching between different geographic map projections for the interpretation of the geographic nature of the data.

As the performance analysis in Section 7.1 showed, the dynamic generation of geometry as part of the rendering process reduces the rendering performance and therefore limits the number of data points that can be processed interactively. However, even on consumer graphics hardware, the approach enables rendering of large data sets, such as 100 000 trajectories or networks with up to 2 million edges, still at interactive frame

rates. On the other hand, update performance is increased significantly, which is crucial for maintaining the interactivity of the approach.

In practice, the described approach will likely be embedded into larger toolsets of diverse visual analytics methods, which combine exploratory visualization with analytical methods and sophisticated algorithmical analysis techniques. These methods will most probably not be calculated as part of the rendering pipeline. Therefore, a GPU-based visualization pipeline as described here would not be implemented in this exact way. Yet, the concept may serve as an important building block for visual analytics tools in the future, which will have to deal with large amounts of possibly real-time, spatial, and time-dependent data.

Over the last decade, the amount of data that is produced regularly and permanently has grown steadily. Examples of this are automatically generated data by traffic surveillance systems, for example for air traffic, vehicle, or pedestrian movements. Other data sources include remote sensing, which scan our physical environments, such as landscapes and cities, and produce large data sets, which will constantly become more detailed and recent. Large sensor networks will produce public and private data in real-time. Also, the systems of the information age will continue to produce large data streams, such as social media interactions and communication events. This development is likely to continue in the future.

While many of these data sources will be analyzed automatically by AI systems, the need for human interpretation and control for these large amounts of data will also grow. In this context, visual analytics systems will play an important role to enable the exploration, analysis, and interpretation of large data sets, and to communicate the results of such analyses.

Therefore, new methods will need to be developed that enable not only the visualization of such large and diverse data sets, but also their interactive analysis. For this, specialized coprocessors, such as GPU hardware, will need to be utilized. At the moment, the development of computer graphics hardware goes mainly into the direction of rendering very large scenes with higher quality, acceleration of ray casting approaches for photorealistic rendering, and most recently the support of machine learning algorithms.

However, the need for interactive visualization and exploration systems will hopefully also increase research and development for utilizing the GPU more directly towards the needs for data processing and visualization. In particular, methods for the management, processing, and visualization of large complex data sets, which are geo-referenced, time-dependent, and will be streamed in real-time, will hopefully be developed in the future. This would enable the development of larger interactive visualization and exploration tools for the increasing amount of available data sets.

References

[1] Flightradar24 AB. *Flight Tracker | Flightradar24 | Track Planes In Real-Time*. URL: https://www.flightradar24.com (visited on 07/10/2018).

[2] Federal Aviation Administration. *Automatic Dependent Surveillance-Broadcast (ADS-B)*. URL: https://www.faa.gov/nextgen/programs/adsb (visited on 07/07/2018).

[3] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-time rendering*. AK Peters/CRC Press, 2018.

[4] Gennady Andrienko and Natalia Andrienko. "Dynamic time transformations for visualizing multiple trajectories in interactive space-time cube". In: *International Cartographic Conference, ICC*. 2011.

[5] Gennady Andrienko and Natalia Andrienko. "Exploration of massive movement data: a visual analytics approach". In: *AGILE'08* (2008).

[6] Gennady Andrienko and Natalia Andrienko. "Exploring Trajectory Attributes in Brest Harbor". In: *COST MOVE Workshop on Moving Objects at Sea*. 2013. URL: http://openaccess.city.ac.uk/2911/.

[7] Gennady Andrienko and Natalia Andrienko. "Interactive cluster analysis of diverse types of spatiotemporal data". In: *Acm Sigkdd Explorations Newsletter* 11.2 (2010), pp. 19–28.

[8] Gennady Andrienko, Natalia Andrienko, Michael Burch, and Daniel Weiskopf. "Visual analytics methodology for eye movement studies". In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pp. 2889–2898.

[9] 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". In: *International Journal of Geographical Information Science* 24.10 (2010), pp. 1577–1600.

[10] Gennady Andrienko, Natalia Andrienko, Jason Dykes, Sara Irina Fabrikant, and Monica Wachowicz. "Geovisualization of Dynamics, Movement and Change: Key Issues and Developing Approaches in Visualization Research". In: *Information Visualization* 7.3/4 (2008), pp. 173–180.

[11] Gennady Andrienko, Natalia Andrienko, Georg Fuchs, and Jose Manuel Cordero Garcia. "Clustering trajectories by relevant parts for air traffic analysis". In: *IEEE transactions on visualization and computer graphics* 24.1 (2018), pp. 34–44.

[12] Gennady Andrienko, Natalia Andrienko, Martin Mladenov, Michael Mock, and Christian Poelitz. "Extracting events from spatial time series". In: *Information Visualisation (IV), 2010 14th International Conference*. IEEE. 2010, pp. 48–53.

[13] Gennady Andrienko, Natalia Andrienko, Salvatore Rinzivillo, Mirco Nanni, Dino Pedreschi, and Fosca Giannotti. "Interactive visual clustering of large collections of trajectories". In: *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*. IEEE. 2009, pp. 3–10.

[14] Gennady Andrienko, Natalia Andrienko, Heidrun Schumann, and Christian Tominski. "Visualization of trajectory attributes in space–time cube and trajectory wall". In: *Cartography from Pole to Pole*. Springer, 2014, pp. 157–163.

[15] Gennady Andrienko, Natalia Andrienko, and Stefan Wrobel. "Visual analytics tools for analysis of movement data". In: *ACM SIGKDD Explorations Newsletter* 9.2 (2007), pp. 38–46.

[16] N. Andrienko, G. Andrienko, and P. Gatalsky. "Exploratory spatio-temporal visualization: an analytical review". In: *Journal of Visual Languages and Computing* 14.6 (Dec. 2003), pp. 503–541. ISSN: 1045926X. DOI: 10.1016/S1045-926X(03)00046-6.

[17] N Andrienko, G Andrienko, and P Gatalsky. "Visual data exploration using space-time cube". In: *Proceedings of the 21st International Cartographic Conference*. 2003, pp. 1981–1983.

[18] N. Andrienko, G. Andrienko, N. Pelekis, and S. Spaccapietra. "Basic Concepts of Movement Data". In: *Mobility, Data Mining and Privacy: Geographic Knowledge Discovery*. Ed. by Fosca Giannotti and Dino Pedreschi. Springer Berlin Heidelberg, 2008, pp. 15–38. ISBN: 978-3-540-75177-9. DOI: 10.1007/978-3-540-75177-9_2.

[19] Natalia Andrienko and Gennady Andrienko. "Building a Visual Summary of Multiple Trajectories". In: *GeoViz Hamburg 2009 Workshop, Hamburg, Mar*. 2009, pp. 3–5.

[20] Natalia Andrienko and Gennady Andrienko. "Spatial generalization and aggregation of massive movement data". In: *IEEE Transactions on visualization and computer graphics* 17.2 (2011), pp. 205–219.

[21] Natalia Andrienko, Gennady Andrienko, Louise Barrett, Marcus Dostie, and Peter Henzi. "Space transformation for understanding group movement". In: *IEEE transactions on visualization and computer graphics* 19.12 (2013), pp. 2169–2178.

[22] Natalia Andrienko, Gennady Andrienko, Jose Manuel Cordero Garcia, and David Scarlatti. "Analysis of Flight Variability: a Systematic Approach". In: *IEEE transactions on visualization and computer graphics* (2018).

[23] Natalia Andrienko, Gennady Andrienko, and Peter Gatalsky. "Towards exploratory visualization of spatio-temporal data". In: *Third AGILE Conference on Geographical Information Science*. 2000, pp. 137–142.

[24] Natalia Andrienko, Gennady Andrienko, Monica Wachowicz, and Daniel Orellana. "Uncovering interactions between moving objects". In: *GIScience, 5th international conference, Proceedings*. 2008, pp. 16–26.

[25] 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". In: *Visual Informatics* 1.1 (2017), pp. 25–39.

[26] Daniel Archambault, Helen Purchase, and Bruno Pinaud. "Animation, small multiples, and the effect of mental map preservation in dynamic graphs". In: *IEEE Transactions on Visualization and Computer Graphics* 17.4 (2011), pp. 539–552.

[27] Benjamin Bach, Emmanuel Pietriga, and Jean-Daniel Fekete. "Visualizing dynamic networks with matrix cubes". In: *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM. 2014, pp. 877–886.

[28] Mike Bailey. "Using GPU shaders for visualization". In: *IEEE Computer Graphics and Applications* 29.5 (2009), pp. 96–100.

[29] Peter Bak, Florian Mansmann, Halldor Janetzko, and Daniel Keim. "Spatiotemporal analysis of sensor logs using growth ring maps". In: *IEEE transactions on visualization and computer graphics* 15.6 (2009), pp. 913–920.

[30] Hannah Bast, Patrick Brosi, and Sabine Storandt. "Real-time movement visualization of public transit data". In: *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM. 2014, pp. 331–340.

[31] Jacques Bertin. "Semiology of graphics: diagrams, networks, maps". In: (1983).

[32] Elisa Bertino, Elena Camossi, and Michela Bertolotto. "Multi-granular spatiotemporal object models: concepts and research directions". In: *International Conference on Object Databases*. Springer. 2009, pp. 132–148.

[33] Alexey Boreskov and Evgeniy Shikin. *Computer graphics: from pixels to programmable graphics hardware*. CRC Press, 2013.

[34] Marc Bourgois, Matthew Cooper, Vu Duong, Jonas Hjalmarsson, Marcus Lange, and Anders Ynnerman. "Interactive and immersive 3D visualization for ATC". In: (2005).

[35] Juri Buchmüller, Halldór Janetzko, Gennady Andrienko, Natalia Andrienko, Georg Fuchs, and Daniel A Keim. "Visual analytics for exploring local impact of air traffic". In: *Computer Graphics Forum*. Vol. 34. 3. Wiley Online Library. 2015, pp. 181–190.

[36] Kai Bürger, Roland Fraedrich, Dorit Merhof, and Rüdiger Westermann. "Instant visitation maps for interactive visualization of uncertain particle trajectories." In: *Visualization and Data Analysis*. 2012, 82940P.

[37] John V. Carlis and Joseph A. Konstan. “Interactive visualization of serial periodic data”. In: *Proceedings of the 11th annual ACM symposium on User interface software and technology*. ACM. 1998, pp. 29–38.

[38] Francesco Carucci and L. Studios. “Inside geometry instancing”. In: *GPU Gems* 2 (2005), pp. 47–67.

[39] Alexandre Carvalho, A. Augusto De Sousa, Cristina Ribeiro, and Emilia Costa. “A temporal focus+ context visualization model for handling valid-time spatial information”. In: *Information Visualization* 7.3-4 (2008), pp. 265–274.

[40] Nicholas R. Chrisman, David J. Cowen, Peter F. Fisher, Michael F. Goodchild, and David M. Mark. “Geographic information systems”. In: *Geography in America* (1989), pp. 353–375.

[41] Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. “BDAM-Batched Dynamic Adaptive Meshes for high performance terrain visualization”. In: *Computer Graphics Forum*. Vol. 22. 3. Wiley Online Library. 2003, pp. 505–514.

[42] Andy Cockburn, Amy Karlson, and Benjamin B Bederson. “A review of overview+ detail, zooming, and focus+ context interfaces”. In: *ACM Computing Surveys (CSUR)* 41.1 (2009), p. 2.

[43] Paolo Compieta, Sergio Di Martino, Michela Bertolotto, Filomena Ferrucci, and T. Kechadi. “Exploratory spatio-temporal data mining and visualization”. In: *Journal of Visual Languages & Computing* 18.3 (2007), pp. 255–279.

[44] Kristin A. Cook and James J. Thomas. *Illuminating the path: The research and development agenda for visual analytics*. Tech. rep. Pacific Northwest National Laboratory (PNNL), Richland, WA (US), 2005.

[45] Noel Cressie and Christopher K. Wikle. *Statistics for spatio-temporal data*. John Wiley & Sons, 2015.

[46] Tarik Crnovrsanin, Chris Muelder, Carlos Correa, and Kwan-Liu Ma. “Proximity-based visualization of movement trace data”. In: *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*. IEEE. 2009, pp. 11–18.

[47] Timothy J. Cullip and Ulrich Neumann. *Accelerating volume reconstruction with 3D texture hardware*. Tech. rep. 1993.

[48] Urska Demsar, Kevin Buchin, E. Emiel van Loon, and Judy Shamoun-Baranes. “Stacked space-time densities: a geovisualisation approach to explore dynamics of space use over time”. In: *GeoInformatica* 19.1 (2015), p. 85.

[49] Urška Demšar and Kirsi Virrantaus. “Space–time density of trajectories: exploring spatio-temporal patterns in movement data”. In: *International Journal of Geographical Information Science* 24.10 (2010), pp. 1527–1542.

[50] David DiBiase, Alan M. MacEachren, John B. Krygier, and Catherine Reeves. “Animation and the role of map design in scientific visualization”. In: *Cartography and geographic information systems* 19.4 (1992), pp. 201–214.

[51] Jonathan F Donges, Hanna CH Schultz, Norbert Marwan, Yong Zou, and Jürgen Kurths. “Investigating the topology of interacting networks”. In: *The European Physical Journal B* 84.4 (2011), pp. 635–651.

[52] Jonathan F Donges, Yong Zou, Norbert Marwan, and Jürgen Kurths. “Complex networks in climate dynamics”. In: *The European Physical Journal Special Topics* 174.1 (2009), pp. 157–179.

[53] Jonathan F Donges, Yong Zou, Norbert Marwan, and Jürgen Kurths. “The backbone of the climate network”. In: *EPL (Europhysics Letters)* 87.4 (2009), p. 48007.

[54] Steve Dübel, Martin Röhlig, Heidrun Schumann, and Matthias Trapp. “2D and 3D presentation of spatial data: a systematic review”. In: *3DVis (3DVis), 2014 IEEE VIS International Workshop on*. IEEE. 2014, pp. 11–18.

[55] Todd Eaglin, Isaac Cho, and William Ribarsky. “Space-Time Kernel Density Estimation for Real-Time Interactive Visual Analytics”. In: *Proceedings of the 50th Hawaii International Conference on System Sciences*. 2017.

[56] Robert Edsall and Donna Peuquet. “A graphical user interface for the integration of time into GIS”. In: *Proceedings of the 1997 American congress of surveying and mapping annual convention and exhibition*. 1997, pp. 182–189.

[57] Willem Eerland, Simon Box, Hans Fangohr, and Andras Sobester. “A Gaussian process based decision support tool for air-traffic management”. In: (2017).

[58] Niklas Elmqvist and Philippas Tsigas. “A Taxonomy of 3D Occlusion Management for Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 14.5 (2008), pp. 1095–1109.

[59] Martin Erwig, Ralf Hartmut Güting, Markus Schneider, and Michalis Vazirgiannis. “Spatio-temporal data types: An approach to modeling and querying moving objects in databases”. In: *GeoInformatica* 3.3 (1999), pp. 269–296.

[60] J. Estes and J. Star. “Geographic information systems”. In: *University of California. Santa Barbara-EEUU. 295p* (1990).

[61] Tianfang B. Fang and Yongmei Lu. “Constructing a Near Real-time Space-time Cube to Depict Urban Ambient Air Pollution Scenario”. In: *Transactions in GIS* 15.5 (2011), pp. 635–649.

[62] James D. Foley, Andries Van Dam, Steven K. Feiner, John F. Hughes, and Richard L. Phillips. *Introduction to computer graphics*. Vol. 55. Reading: Addison-Wesley, 1994.

[63] Ying-Huey Fua, Matthew O Ward, and Elke A Rundensteiner. “Hierarchical parallel coordinates for exploration of large datasets”. In: *Proceedings of the conference on Visualization’99: celebrating ten years*. IEEE Computer Society Press. 1999, pp. 43–50.

[64] Peter Gatalsky, Natalia Andrienko, and Gennady Andrienko. “Interactive analysis of event data using space-time cube”. In: *Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on*. IEEE. 2004, pp. 145–152.

[65] Andrew S. Glassner. *An introduction to ray tracing*. Elsevier, 1989.

[66] Tiago Gonçalves, Ana Paula Afonso, and Bruno Martins. “Visualizing human trajectories: comparing space-time cubes and static maps”. In: *Proceedings of the 28th International BCS Human Computer Interaction Conference on HCI 2014-Sand, Sea and Sky-Holiday HCI*. BCS. 2014, pp. 207–212.

[67] Sebastian Grottel, Guido Reina, and Thomas Ertl. “Optimized data transfer for time-dependent, GPU-based glyphs”. In: *Visualization Symposium, 2009. PacificVis ’09. IEEE Pacific*. IEEE. 2009, pp. 65–72.

[68] Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. “Movement patterns in spatio-temporal data”. In: *Encyclopedia of GIS*. Springer, 2008, pp. 726–732.

[69] Hanqi Guo, Zuchao Wang, Bowen Yu, Huijing Zhao, and Xiaoru Yuan. “Tripvista: Triple perspective visual trajectory analytics and its application on microscopic traffic data at a road intersection”. In: *Pacific Visualization Symposium (PacificVis), 2011 IEEE*. IEEE. 2011, pp. 163–170.

[70] Amarnath Gupta. “Multidimensional Data Formats”. In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Springer US, 2009, pp. 1776–1777. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_1309. URL: https://doi.org/10.1007/978-0-387-39940-9_1309.

[71] Robert B. Haber and David A. McNabb. “Visualization idioms: A conceptual model for scientific visualization systems”. In: *Visualization in scientific computing* 74 (1990), p. 93.

[72] T. Hägerstrand. “What about people in regional science?” In: *Papers of the Regional Science Association* (1970), pp. 7–21.

[73] Susan Havre, Elizabeth Hetzler, Paul Whitney, and Lucy Nowell. “Themeriver: Visualizing thematic changes in large document collections”. In: *IEEE transactions on visualization and computer graphics* 8.1 (2002), pp. 9–20.

[74] K. Priyantha Hewagamage, Masahito Hirakawa, and Tadao Ichikawa. “Interactive visualization of spatiotemporal patterns using spirals on a geographical map”. In: *Visual languages, 1999. Proceedings. 1999 IEEE symposium on*. IEEE. 1999, pp. 296–303.

[75] Harry Hochheiser and Ben Shneiderman. “Dynamic query tools for time series data sets: timebox widgets for interactive exploration”. In: *Information Visualization* 3.1 (2004), pp. 1–18.

[76] Danny Holten and Jarke J. Van Wijk. “Force-Directed Edge Bundling for Graph Visualization”. In: *Computer Graphics Forum* 28.3 (2009), pp. 983–990. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2009.01450.x.

[77] Hugues Hoppe. “Smooth view-dependent level-of-detail control and its application to terrain rendering”. In: *Visualization'98. Proceedings*. IEEE. 1998, pp. 35–42.

[78] Otto Huisman, Irvin Feliciano Santiago, Bas Retsios, and Menno-Jan Kraak. “Development of a geovisual analytics environment for investigating archaeological events based upon the space-time Cube”. In: *Extended abstract for GIScience Geovisual Analytics Workshop*. Vol. 23. 2008.

[79] Christophe Hurter, Richard Alligier, David Gianazza, Stéphane Puechmorel, Gennady Andrienko, and Natalia Andrienko. “Wind parameters extraction from aircraft trajectories”. In: *Computers, Environment and Urban Systems* 47 (2014), pp. 28–43.

[80] Christophe Hurter, Stéphane Conversy, David Gianazza, and AC Telea. “Interactive image-based information visualization for aircraft trajectory analysis”. In: *Transportation Research Part C: Emerging Technologies* 47 (2014), pp. 207–227.

[81] Christophe Hurter, Ozan Ersoy, Sara Irina Fabrikant, Tijmen R Klein, and Alexandru C Telea. “Bundled visualization of dynamicgraph and trail data”. In: *IEEE transactions on visualization and computer graphics* 20.8 (2014), pp. 1141–1157.

[82] Christophe Hurter, Ozan Ersoy, and Alexandru Telea. “Graph Bundling by Kernel Density Estimation”. In: *Computer Graphics Forum* 31.3pt1 (June 2012), pp. 865–874. ISSN: 0167-7055. DOI: 10.1111/j.1467-8659.2012.03079.x.

[83] Christophe Hurter, Ozan Ersoy, and Alexandru Telea. “Smooth bundling of large streaming and sequence graphs”. In: *Visualization Symposium (PacificVis), 2013 IEEE Pacific*. IEEE. 2013, pp. 41–48.

[84] Christophe Hurter, Stéphane Puechmorel, Florence Nicol, and Alexandru Telea. “Functional Decomposition for Bundled Simplification of Trail Sets”. In: *IEEE transactions on visualization and computer graphics* 24.1 (2018), pp. 500–510.

[85] Christophe Hurter, Nathalie Henry Riche, Steven M Drucker, Maxime Cordeil, Richard Alligier, and Romain Vuillemot. “FiberClay: Sculpting Three Dimensional Trajectories to Reveal Structural Insights”. In: *IEEE transactions on visualization and computer graphics* (2018).

[86] Christophe Hurter, Mathieu Serrurier, Roland Alonso, Gilles Tabart, and Jean-Luc Vinot. “An automatic generation of schematic maps to display flight routes for air traffic controllers: structure and color optimization”. In: *Proceedings of the international conference on advanced visual interfaces*. ACM. 2010, pp. 233–240.

[87] Christophe Hurter, Benjamin Tissoires, and Stéphane Conversy. “Fromdady: Spreading aircraft trajectories across views to support iterative queries”. In: *IEEE transactions on visualization and computer graphics* 15.6 (2009), pp. 1017–1024.

[88] ICAO. *International Civil Aviation Organization*. URL: https://www.icao.int (visited on 07/07/2018).

[89] ISO/TC 211 Geographic information/Geomatics. *Multi-Lingual Glossary of Terms*. http://www.isotc211.org/Terminology.htm. Accessed: 2017-10-31.

[90] A Inselberg and Bernard Dimsdale. "Parallel coordinates: a tool for visualizing multi-dimensional geometry.(1990)". In: (1990).

[91] Alfred Inselberg and Bernard Dimsdale. "Parallel coordinates for visualizing multi-dimensional geometry". In: *Computer Graphics 1987*. Springer, 1987, pp. 25–44.

[92] Victoria Interrante. "Harnessing natural textures for multivariate visualization". In: *IEEE Computer Graphics and Applications* 20.6 (2000), pp. 6–11.

[93] Firdaus Janoos, Shantanu Singh, Okan Irfanoglu, Raghu Machiraju, and Richard Parent. "Activity analysis using spatio-temporal trajectory volumes in surveillance applications". In: *Visual Analytics Science and Technology, 2007. VAST 2007. IEEE Symposium on*. IEEE. 2007, pp. 3–10.

[94] Thomas Kapler and William Wright. "GeoTime information visualization". In: *Information visualization* 4.2 (2005), pp. 136–146.

[95] Arie Kaufman. "Volume visualization". In: *The visual computer* 6.1 (1990).

[96] Daniel A. Keim. "Information visualization and visual data mining". In: *IEEE transactions on Visualization and Computer Graphics* 8.1 (2002), pp. 1–8.

[97] Daniel Keim, Gennady Andrienko, Jean-Daniel Fekete, Carsten Görg, Jörn Kohlhammer, and Guy Melançon. "Visual analytics: Definition, process, and challenges". In: *Information visualization*. Springer, 2008, pp. 154–175.

[98] Andreas Kjellin, Lars Winkler Pettersson, Stefan Seipel, and Mats Lind. "Different levels of 3D: An evaluation of visualized discrete spatiotemporal data in space-time cubes". In: *Information Visualization* 9.2 (2010), pp. 152–164.

[99] Tijmen Klein, Matthew Van Der Zwan, and Alexandru Telea. "Dynamic multi-scale visualization of flight data". In: *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*. Vol. 1. IEEE. 2014, pp. 104–114.

[100] Menno-Jan Kraak. "Geovisualization illustrated". In: *ISPRS journal of photogrammetry and remote sensing* 57.5 (2003), pp. 390–399.

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

[102] Menno-Jan Kraak. "Timelines, temporal resolution, temporal zoom and time geography". In: *Proceedings 22nd International Cartographic Conference, A Coruna Spain*. 2005.

[103] Menno-Jan Kraak, Rob Edsall, and Alan M. MacEachren. "Cartographic animation and legends for temporal maps: Exploration and or interaction". In: *Proceedings of the 18th International Cartographic Conference*. Vol. 1. 1997, pp. 253–261.

[104] Menno-Jan Kraak and Nannan He. “Organizing the neo-geography collections with annotated space-time paths”. In: *The 24th International Cartographic Conference, Chile*. 2009.

[105] Menno-Jan Kraak and Alexandra Koussoulakou. “A visualization environment for the space-time-cube”. In: *Developments in spatial data handling* (2005), pp. 189–200.

[106] Menno-Jan Kraak and P. F. Madzudzo. “Space time visualization for epidemiological research”. In: *ICC 2007: Proceedings of the 23nd international cartographic conference ICC: Cartography for everyone and for you*. International Cartographic Association. 2007.

[107] Per Ola Kristensson, Nils Dahlback, Daniel Anundi, Marius Bjornstad, Hanna Gillberg, Jonas Haraldsson, Ingrid Martensson, Mathias Nordvall, and Josefine Stahl. “An evaluation of space time cube representation of spatiotemporal patterns”. In: *IEEE Transactions on Visualization and Computer Graphics* 15.4 (2009), pp. 696–702.

[108] Per Ola Kristensson, Nils Dahlback, Daniel Anundi, Marius Bjornstad, Hanna Gillberg, Jonas Haraldsson, Ingrid Martensson, Matttias Nordvall, and Josefin Stahl. “The trade-offs with space time cube representation of spatiotemporal patterns”. In: *arXiv preprint arXiv:0707.1618* (2007).

[109] Michael Krone, Katrin Bidmon, and Thomas Ertl. “GPU-based Visualisation of Protein Secondary Structure.” In: *TPCG* 8 (2008), pp. 115–122.

[110] Michael Krone, Katrin Bidmon, and Thomas Ertl. “Interactive visualization of molecular surface dynamics”. In: *IEEE transactions on visualization and computer graphics* 15.6 (2009), pp. 1391–1398.

[111] Michael Krone, Martin Falk, Sascha Rehm, Jürgen Pleiss, and Thomas Ertl. “Interactive exploration of protein cavities”. In: *Computer Graphics Forum*. Vol. 30. 3. Wiley Online Library. 2011, pp. 673–682.

[112] Jens Krüger and Rüdiger Westermann. “Acceleration techniques for GPU-based volume rendering”. In: *Proceedings of the 14th IEEE Visualization 2003 (VIS03)*. IEEE Computer Society. 2003, p. 38.

[113] Robert Krüger, Dennis Thom, Michael Wörner, Harald Bosch, and Thomas Ertl. “TrajectoryLenses–A Set-based Filtering and Exploration Technique for Long-term Trajectory Data”. In: *Computer Graphics Forum*. Vol. 32. 3pt4. Wiley Online Library. 2013, pp. 451–460.

[114] Kuno Kurzhals and Daniel Weiskopf. “Space-time visual analytics of eye-tracking data for dynamic stimuli”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2129–2138.

[115] Irma Kveladze and Menno-Jan Kraak. “What do we know about the space-time cube from cartographic and usability perspective”. In: *Columbus, Ohio, USA: Proceedings of Autocarto* (2012), pp. 16–18.

[116] Irma Kveladze, Menno-Jan Kraak, and C. P. van Elzakker. “A methodological framework for researching the usability of the space-time cube”. In: *The Cartographic Journal* 50.3 (2013), pp. 201–210.

[117] Irma Kveladze, Menno-Jan Kraak, and C. P. Van Elzakker. “The space-time cube as part of a GeoVisual analytics environment to support the understanding of movement data”. In: *International Journal of Geographical Information Science* 29.11 (2015), pp. 2001–2016.

[118] Ove Daae Lampe, Johannes Kehrer, and Helwig Hauser. “Visual analysis of multivariate movement data using interactive difference views.” In: *VMV*. Vol. 10. 2010, pp. 315–322.

[119] Ove Daae Lampe, Ivan Viola, Nathalie Reuter, and Helwig Hauser. “Two-level approach to efficient visualization of protein dynamics”. In: *IEEE transactions on visualization and computer graphics* 13.6 (2007), pp. 1616–1623.

[120] Devin Lange, Francesca Samsel, Ioannis Karamouzas, S. J. Guy, Rodney Dockter, Timothy Kowalewski, and Daniel F. Keefe. “Trajectory Mapper: Interactive Widgets and Artist-Designed Encodings for Visualizing Multivariate Trajectory Data”. In: *EuroVis 2017 - Short Papers*. Ed. by Barbora Kozlikova, Tobias Schreck, and Thomas Wischgoll. The Eurographics Association, 2017. ISBN: 978-3-03868-043-7. DOI: 10.2312/eurovisshort.20171141.

[121] Xia Li, Arzu Çöltekin, and Menno-Jan Kraak. “Visual exploration of eye movement data using the space-time-cube”. In: *International Conference on Geographic Information Science*. Springer. 2010, pp. 295–309.

[122] Xia Li and Menno-Jan Kraak. “The time wave. A new method of visual exploration of geo-data in time–space”. In: *The Cartographic Journal* 45.3 (2008), pp. 193–200.

[123] Zhicheng Liu, Biye Jiang, and Jeffrey Heer. “imMens: Real-time Visual Querying of Big Data”. In: *Computer Graphics Forum*. Vol. 32. 3pt4. Wiley Online Library. 2013, pp. 421–430.

[124] Yotam Livny, Zvi Kogan, and Jihad El-Sana. “Seamless patches for GPU-based terrain rendering”. In: *The Visual Computer* 25.3 (2009), pp. 197–208.

[125] Frank Losasso and Hugues Hoppe. “Geometry clipmaps: terrain rendering using nested regular grids”. In: *ACM Transactions on Graphics (TOG)*. Vol. 23. 3. ACM. 2004, pp. 769–776.

[126] Martin Luboschik, Martin Röhlig, Arne T Bittig, Natalia Andrienko, Heidrun Schumann, and Christian Tominski. “Feature-Driven Visual Analytics of Chaotic Parameter-Dependent Movement”. In: *Computer Graphics Forum*. Vol. 34. 3. Wiley Online Library. 2015, pp. 421–430.

[127] David P. Luebke. *Level of detail for 3D graphics*. Morgan Kaufmann, 2003.

[128] Frank Luna. *Introduction to 3D Game Programming with DirectX 11*. Mercury Learning & Information, 2012. ISBN: 1936420228, 9781936420223.

[129] Alan M. MacEachren. *How maps work: representation, visualization, and design*. Guilford Press, 1995.

[130] Sara Maggi, Sara Irina Fabrikant, Jean-Paul Imbert, and Christophe Hurter. "How do display design and user characteristics matter in animations? An empirical study with air traffic control displays". In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 51.1 (2016), pp. 25–37.

[131] Patrick S. McCormick, Jeff Inman, James P. Ahrens, Charles Hansen, and Greg Roth. "Scout: A hardware-accelerated system for quantitatively driven visualization and analysis". In: *Proceedings of the conference on Visualization'04*. IEEE Computer Society. 2004, pp. 171–178.

[132] Bryan McDonnel and Niklas Elmqvist. "Towards utilizing gpus in information visualization: A model and implementation of image-space operations". In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 1105–1112.

[133] Jennis Meyer-Spradow, Timo Ropinski, Jörg Mensmann, and Klaus Hinrichs. "Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations". In: *IEEE Computer Graphics and Applications* 29.6 (2009), pp. 6–13.

[134] Peng Mi, Maoyuan Sun, Moeti Masiane, Yong Cao, and Chris North. "AVIST: A GPU-Centric Design for Visual Exploration of Large Multidimensional Datasets". In: *Informatics*. Vol. 3. 4. Multidisciplinary Digital Publishing Institute. 2016, p. 18.

[135] Mohamed F. Mokbel, Thanaa M. Ghanem, and Walid G. Aref. "Spatio-temporal access methods". In: *IEEE Data Eng. Bull.* 26.2 (2003), pp. 40–49.

[136] Nora Molkenthin, Kira Rehfeld, Norbert Marwan, and Jürgen Kurths. "Networks from flows-from dynamics to topology". In: *Scientific reports* 4 (2014), p. 4119.

[137] Aaftab Munshi, Dan Ginsburg, and Dave Shreiner. *The OpenGL ES 2.0 programming guide*. Addison-Wesley, 2009. ISBN: 9780321502797 0321502795.

[138] Tamara Munzner. "Visualization". In: *Fundamentals of Graphics*. 2009. Chap. 27, pp. 675–707.

[139] Tomoki Nakaya and Keiji Yano. "Visualising Crime Clusters in a Space-time Cube: An Exploratory Data-analysis Approach Using Space-time Kernel Density Estimation and Scan Statistics". In: *Transactions in GIS* 14.3 (2010), pp. 223–239.

[140] Marc Nienhaus and Jürgen Döllner. "Depicting dynamics using principles of visual art and narrations". In: *IEEE Computer Graphics and Applications* 25.3 (2005), pp. 40–51.

[141] John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips. "GPU computing". In: *Proceedings of the IEEE* 96.5 (2008), pp. 879–899.

[142] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Timothy J. Purcell. "A survey of general-purpose computation on graphics hardware". In: *Computer graphics forum*. Vol. 26. 1. Wiley Online Library. 2007, pp. 80–113.

[143] Christine Parent, Stefano Spaccapietra, and Esteban Zimányi. "Spatio-temporal conceptual models: data structures + space + time". In: *Proceedings of the 7th ACM international symposium on Advances in geographic information systems*. ACM. 1999, pp. 26–33.

[144] Stefan Peters and Jukka M. Krisp. "Density calculation for moving points". In: *Proceeding of the 13th AGILE international conference on geographic information science, Guimaraes, Portugal*. Vol. 1014. 2010.

[145] Vid Petrovic, James Fallon, and Falko Kuester. "Visualizing whole-brain DTI tractography with GPU-based tuboids and LoD management". In: *IEEE transactions on visualization and computer graphics* 13.6 (2007), pp. 1488–1495.

[146] Vsevolod Peysakhovich, Christophe Hurter, and Alexandru Telea. "Attribute-driven edge bundling for general graphs with applications in trail analysis". In: *Visualization Symposium (PacificVis), 2015 IEEE Pacific*. IEEE. 2015, pp. 39–46.

[147] Stanislav Popelka and Vit Voženilek. "Specifying of requirements for spatio-temporal data in map by eye-tracking and space-time-cube". In: *International Conference on Graphic and Image Processing (ICGIP 2012)*. 2013.

[148] Donghao Ren, Bongshin Lee, and Tobias Höllerer. "Stardust: Accessible and Transparent GPU Support for Information Visualization Rendering". In: *Computer Graphics Forum*. Vol. 36. 3. Wiley Online Library. 2017, pp. 179–188.

[149] Christophe Riccio and Sean Lilley. "Introducing the programmable vertex pulling rendering pipeline". In: *GPU Pro*. Ed. by Wolfgang Engel. 4th ed. CRC Press, Apr. 2013.

[150] Salvatore Rinzivillo, Dino Pedreschi, Mirco Nanni, Fosca Giannotti, Natalia Andrienko, and Gennady Andrienko. "Visually driven analysis of movement data by progressive clustering". In: *Information Visualization* 7.3-4 (2008), pp. 225–239.

[151] Jonathan C. Roberts. "Exploratory visualization with multiple linked views". In: *Exploring Geovisualization*. Ed. by Jason Dykes, Alan M. MacEachren, and Menno-Jan Kraak. Elsevier, 2004, pp. 149–170.

[152] George Robertson, Roland Fernandez, Danyel Fisher, Bongshin Lee, and John Stasko. "Effectiveness of animation in trend visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008).

[153] John F. Roddick and Myra Spiliopoulou. "A bibliography of temporal, spatial and spatio-temporal data mining research". In: *ACM SIGKDD Explorations Newsletter* 1.1 (1999), pp. 34–38.

[154] Ruth Rosenholtz, Yuanzhen Li, and Lisa Nakano. "Measuring visual clutter". In: *Journal of vision* 7.2 (2007), pp. 17–17.

[155] Szymon Rusinkiewicz and Marc Levoy. "QSplat: A multiresolution point rendering system for large meshes". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 2000, pp. 343–352.

[156] Roeland Scheepens, Christophe Hurter, Huub Van De Wetering, and Jarke J. Van Wijk. "Visualization, selection, and analysis of traffic flows". In: *IEEE transactions on visualization and computer graphics* 22.1 (2016), pp. 379–388.

[157] Roeland Scheepens, Niels Willems, Huub Van de Wetering, Gennady Andrienko, Natalia Andrienko, and Jarke J. Van Wijk. "Composite density maps for multivariate trajectories". In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2518–2527.

[158] Roeland Scheepens, Niels Willems, Huub van de Wetering, and Jarke J. Van Wijk. "Interactive density maps for moving objects". In: *IEEE Computer Graphics and Applications* 32.1 (2012), pp. 56–66.

[159] Roeland Scheepens, Niels Willems, Huub van de Wetering, and Jarke J. Van Wijk. "Interactive visualization of multivariate trajectory data with density maps". In: *Pacific Visualization Symposium (PacificVis), 2011 IEEE*. IEEE. 2011, pp. 147–154.

[160] Marc Schirski, Torsten Kuhlen, Martin Hopp, Philipp Adomeit, Stefan Pischinger, and Christian Bischof. "Efficient visualization of large amounts of particle trajectories in virtual environments using virtual tubelets". In: *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*. ACM. 2004, pp. 141–147.

[161] Jens Schneider and Rüdiger Westermann. "GPU-friendly high-quality terrain rendering". In: (2006).

[162] Will J Schroeder, Bill Lorensen, and Ken Martin. *The visualization toolkit: an object-oriented approach to 3D graphics*. Kitware, 2004.

[163] G. Sellers and J.M. Kessenich. *Vulkan Programming Guide: The Official Guide to Learning Vulkan*. Always learning. Addison Wesley, 2016. ISBN: 9780134464541.

[164] Shih-Lung Shaw, Hongbo Yu, and Leonard S. Bombom. "A space-time GIS approach to exploring large individual-based spatiotemporal datasets". In: *Transactions in GIS* 12.4 (2008), pp. 425–441.

[165] Ben Shneiderman. "Dynamic queries for visual information seeking". In: *IEEE software* 11.6 (1994), pp. 70–77.

[166] Dave Shreiner and The Khronos OpenGL ARB Working Group. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1*. 7th. Addison-Wesley Professional, 2009. ISBN: 0321552628, 9780321552624.

[167] Dave Shreiner, Graham Sellers, John Kessenich, and Bill Licea-Kane. "OpenGL programming guide: The Official guide to learning OpenGL, version 4.3". In: Addison-Wesley, 2013. Chap. 10, pp. 509–561.

[168] Dave Shreiner, Graham Sellers, John Kessenich, and Bill Licea-Kane. “OpenGL programming guide: The Official guide to learning OpenGL, version 4.3”. In: Addison-Wesley, 2013. Chap. 12, pp. 623–648.

[169] Dave Shreiner, Graham Sellers, John Kessenich, and Bill Licea-Kane. “OpenGL programming guide: The Official guide to learning OpenGL, version 4.3”. In: Addison-Wesley, 2013. Chap. 6, pp. 319–320.

[170] Christian Sigg, Tim Weyrich, Mario Botsch, and Markus H Gross. “GPU-based ray-casting of quadratic surfaces.” In: *Eurographics Symposium on Point-Based Graphics*. 2006, pp. 59–65.

[171] Merrill Ivan Skolnik. *Radar Handbook*. Electronic engineering series. McGraw-Hill, 1990. ISBN: 9780070579132.

[172] Aidan Slingsby, Jo Wood, and Jason Dykes. “Treemap cartography for showing spatial and temporal traffic patterns”. In: *Journal of Maps* 6.1 (2010), pp. 135–146.

[173] Marc Stamminger and George Drettakis. “Interactive sampling and rendering for complex and procedural geometry”. In: *Rendering Techniques 2001*. Springer, 2001, pp. 151–162.

[174] Simon Stegmaier, Magnus Strengert, Thomas Klein, and Thomas Ertl. “A simple and flexible volume rendering framework for graphics-hardware-based raycasting”. In: *Volume Graphics, 2005. Fourth International Workshop on*. IEEE. 2005, pp. 187–241.

[175] Carsten Stoll, Stefan Gumhold, and H.-P. Seidel. “Visualization with stylized line primitives”. In: *Visualization, 2005. VIS 05. IEEE*. IEEE. 2005, pp. 695–702.

[176] Magnus Strengert, Marcelo Magallón, Daniel Weiskopf, Stefan Guthe, and Thomas Ertl. “Hierarchical visualization and compression of large volume datasets using GPU clusters.” In: *EGPGV*. 2004, pp. 41–48.

[177] Alexandru Telea and Robert Strzodka. “Multiscale image based flow visualization”. In: *Visualization and Data Analysis*. 2006, p. 606001.

[178] Sidharth Thakur and Andrew J. Hanson. “A 3D visualization of multiple time series on maps”. In: *Information Visualisation (IV), 2010 14th International Conference*. IEEE. 2010, pp. 336–343.

[179] Christian Tominski, James Abello, and Heidrun Schumann. “Axes-based visualizations with radial layouts”. In: *Proceedings of the 2004 ACM symposium on Applied computing*. ACM. 2004, pp. 1242–1247.

[180] Christian Tominski and Hans-Joerg Schulz. “The great wall of space-time”. In: *Proceedings of the Workshop on Vision, Modeling & Visualization*. The Eurographics Association, 2012, pp. 199–206.

[181] Christian Tominski, Petra Schulze-Wollgast, and Heidrun Schumann. “3d information visualization for time dependent data on maps”. In: *Information Visualisation, 2005. Proceedings. Ninth International Conference on*. IEEE. 2005, pp. 175–181.

[182] Christian Tominski and Heidrun Schumann. “Enhanced interactive spiral display”. In: *SIGRAD 2008. The Annual SIGRAD Conference Special Theme: Interaction; November 27-28; 2008 Stockholm; Sweden*. 034. Linköping University Electronic Press. 2008, pp. 53–56.

[183] Christian Tominski, Heidrun Schumann, Gennady Andrienko, and Natalia Andrienko. “Stacking-based visualization of trajectory attribute data”. In: *IEEE Transactions on visualization and Computer Graphics* 18.12 (2012), pp. 2565–2574.

[184] Melanie Tory, Arthur E Kirkpatrick, M Stella Atkins, and Torsten Moller. “Visualization task performance with 2D, 3D, and combination displays”. In: *IEEE transactions on visualization and computer graphics* 12.1 (2006), pp. 2–13.

[185] Matthias Trapp. “Interactive rendering techniques for focus+ context visualization of 3d geovirtual environments”. PhD thesis. Universitätsbibliothek der Universität Potsdam, 2013.

[186] Nectaria Tryfona and Dieter Pfoser. “Designing ontologies for moving objects applications”. In: *Proc. of the International Workshop on Complex Reasoning on Geographic Data, Paphos, Cyprus*. 2001.

[187] Anastasios A Tsonis and Paul J Roebber. “The architecture of the climate network”. In: *Physica A: Statistical Mechanics and its Applications* 333 (2004), pp. 497–504.

[188] Liubov Tupikina, Kira Rehfeld, Nora Molkenthin, Veronika Stolbova, Norbert Marwan, and Jürgen Kurths. “Characterizing the evolution of climate networks”. In: *Nonlinear Processes in Geophysics* (2014).

[189] Barbara Tversky, Julie Bauer Morrison, and Mireille Betrancourt. “Animation: can it facilitate?” In: *International journal of human-computer studies* 57.4 (2002), pp. 247–262.

[190] Allen Van Gelder and Kwansik Kim. “Direct volume rendering with shading via three-dimensional textures”. In: *Proceedings of the 1996 symposium on Volume visualization*. IEEE Press. 1996, 23–ff.

[191] Michael Vassilakopoulos and Antonio Corral. “Spatio-Temporal Indexing Techniques”. In: *Handbook of Research on Innovations in Database Technologies and Applications: Current and Future Trends*. IGI Global, 2009, pp. 260–268.

[192] Zuchao Wang, Tangzhi Ye, Min Lu, Xiaoru Yuan, Huamin Qu, Jacky Yuan, and Qianliang Wu. “Visual exploration of sparse traffic trajectory data”. In: *IEEE transactions on visualization and computer graphics* 20.12 (2014), pp. 1813–1822.

[193] Matthew O. Ward. “A taxonomy of glyph placement strategies for multidimensional data visualization”. In: *Information Visualization* 1.3-4 (2002), pp. 194–210.

[194] Colin Ware. “3D contour perception for flow visualization”. In: *Proceedings of the 3rd symposium on Applied perception in graphics and visualization*. ACM. 2006, pp. 101–106.

[195] Colin Ware. *Information visualization: perception for design*. Elsevier, 2012.

[196] Colin Ware, Roland Arsenault, Matthew Plumlee, and David Wiley. "Visualizing the underwater behavior of humpback whales". In: *IEEE Computer Graphics and Applications* 26.4 (2006), pp. 14–18.

[197] Colin Ware, Daniel Bolan, Ricky Miller, David H. Rogers, and James P. Ahrens. "Animated versus static views of steady flow patterns". In: *Proceedings of the ACM Symposium on Applied Perception*. ACM. 2016, pp. 77–84.

[198] Chris Weaver. "Cross-filtered views for multidimensional visual analysis". In: *IEEE Transactions on Visualization and Computer Graphics* 16.2 (2010), pp. 192–204.

[199] Marc Weber, Marc Alexa, and Wolfgang Müller. "Visualizing time-series on spirals." In: *Infovis*. Vol. 1. 2001, pp. 7–14.

[200] Jarke J. van Wijk. "Image based flow visualization". In: *ACM Transactions on Graphics (ToG)* 21.3 (2002), pp. 745–754.

[201] Niels Willems, Huub Van De Wetering, and Jarke J. Van Wijk. "Evaluation of the visibility of vessel movement features in trajectory visualizations". In: *Computer Graphics Forum*. Vol. 30. 3. Wiley Online Library. 2011, pp. 801–810.

[202] Niels Willems, Huub Van De Wetering, and Jarke J. Van Wijk. "Visualization of vessel movements". In: *Computer Graphics Forum*. Vol. 28. 3. Wiley Online Library. 2009, pp. 959–966.

[203] Niels Willems, Huub van de Wetering, and Jarke J. van Wijk. "Interactive Poster: Visualization of vessel trajectories for maritime safety and security systems". In: *IEEE Information Visualization Conference (InfoVis 2008)*. 2008.

[204] Markus Wolff and Hartmut Asche. "Geovisualization Approaches for Spatio-temporal Crime Scene Analysis – Towards 4D Crime Mapping". In: *Computational Forensics*. Ed. by Zeno J. M. H. Geradts, Katrin Y. Franke, and Cor J. Veenman. Springer Berlin Heidelberg, 2009, pp. 78–89. ISBN: 978-3-642-03521-0.

[205] Jo Wood, Jason Dykes, and Aidan Slingsby. "Visualisation of origins, destinations and flows with OD maps". In: *The Cartographic Journal* 47.2 (2010), pp. 117–129.

[206] Jo Wood, Jason Dykes, Aidan Slingsby, and Robert Radburn. "Flow trees for exploring spatial trajectories". In: *Proceedings of GISR UK*. 2009, pp. 31–34.

[207] Kazuko Yamasaki, Avi Gozolchiani, and Shlomo Havlin. "Climate networks around the globe are significantly affected by El Nino". In: *Physical review letters* 100.22 (2008), p. 228501.

[208] Yalong Yang, Tim Dwyer, Bernhard Jenny, Kim Marriott, Maxime Cordeil, and Haohui Chen. "Origin-Destination Flow Maps in Immersive Environments". In: *IEEE transactions on visualization and computer graphics* (2018).

Publication Overview

[1] Stefan Buschmann and Jürgen Döllner. “Dichte-und Distanzanalyse massiver raumzeitlicher Bewegungsdaten”. In: *Geoinformatik 2012 - Mobilität und Umwelt*. 2012, pp. 67–74.

[2] Stefan Buschmann, Thomas Nocke, Christian Tominski, and Jürgen Döllner. “Towards visualizing geo-referenced climate networks”. In: *Proceedings of Workshop GeoViz Hamburg*. 2013.

[3] Stefan Buschmann, Matthias Trapp, and Jürgen Döllner. “Animated visualization of spatial-temporal trajectory data for air-traffic analysis”. In: *The Visual Computer* 32.3 (2016), pp. 371–381.

[4] Stefan Buschmann, Matthias Trapp, and Jürgen Döllner. “Real-time animated visualization of massive air-traffic trajectories”. In: *Cyberworlds (CW), 2014 International Conference on*. IEEE. 2014, pp. 174–181.

[5] Stefan Buschmann, Matthias Trapp, and Jürgen Döllner. “Real-Time Visualization of Massive Movement Data in Digital Landscapes”. In: *Digital Landscape Architecture*. 2015.

[6] Stefan Buschmann, Matthias Trapp, Patrick Lühne, and Jürgen Döllner. “Hardware-accelerated attribute mapping for interactive visualization of complex 3D trajectories”. In: *Information Visualization Theory and Applications (IVAPP), 2014 International Conference on*. IEEE. 2014, pp. 356–363.

[7] Thomas Nocke, Stefan Buschmann, Jonathan F. Donges, Norber Marwan, Hans-Jörg Schulz, and Christian Tominski. “Review: visual analytics of climate networks”. In: *Nonlinear Processes in Geophysics* 22.5 (2015), p. 545.

[8] Willy Scheibel, Stefan Buschmann, Matthias Trapp, and Jürgen Döllner. “Attributed Vertex Clouds”. In: *GPU Zen*. Ed. by Christopher Oat. 8th ed. GPU Pro. Wolfgang Engel, Mar. 2017.

List of Figures

Listings