RESEARCH ARTICLE | APRIL 14 2023

GPU-based, interactive exploration of large spatiotemporal climate networks

Special Collection: Theory-informed and Data-driven Approaches to Advance Climate Sciences

Stefan Buschmann; Peter Hoffmann; Ankit Agarwal; ... et. al

Chaos 33, 043129 (2023)
https://doi.org/10.1063/5.0131933

GPU-based, interactive exploration of large spatiotemporal climate networks

Cite as: Chaos **33**, 043129 (2023); doi: 10.1063/5.0131933
Submitted: 25 October 2022 · Accepted: 27 March 2023 ·
Published Online: 14 April 2023

Stefan Buschmann,[1] Peter Hoffmann,[2] Ankit Agarwal,[3,a)] Norbert Marwan,[2] and Thomas Nocke[2,b)]

AFFILIATIONS

[1] Hasso Plattner Institute, University of Potsdam, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
[2] Potsdam Institute for Climate Impact Research, Telegrafenberg, 14473 Potsdam, Germany
[3] Department of Hydrology, Indian Institute of Technology Roorkee, Haridwar Highway, Roorkee, Uttarakhand 247667, India

Note: This article is part of the Focus Issue, Theory-informed and Data-driven Approaches to Advance Climate Sciences.
[a)] **Also at:** GFZ German Research Centre for Geosciences, Telegrafenberg, 14473 Potsdam, Germany
[b)] **Author to whom correspondence should be addressed:** nocke@pik-potsdam.de

ABSTRACT

This paper introduces the Graphics Processing Unit (GPU)-based tool Geo-Temporal eXplorer (GTX), integrating a set of highly interactive techniques for visual analytics of large geo-referenced complex networks from the climate research domain. The visual exploration of these networks faces a multitude of challenges related to the geo-reference and the size of these networks with up to several million edges and the manifold types of such networks. In this paper, solutions for the interactive visual analysis for several distinct types of large complex networks will be discussed, in particular, time-dependent, multi-scale, and multi-layered ensemble networks. Custom-tailored for climate researchers, the GTX tool supports heterogeneous tasks based on interactive, GPU-based solutions for on-the-fly large network data processing, analysis, and visualization. These solutions are illustrated for two use cases: multi-scale climatic process and climate infection risk networks. This tool helps one to reduce the complexity of the highly interrelated climate information and unveils hidden and temporal links in the climate system, not available using standard and linear tools (such as empirical orthogonal function analysis).

Teleconnection analysis of climate data is an established research field analyzing network interactions in the climate system. Furthermore, the investigation over types of networks such as electricity, trading, or flight become more into focus in the context of climate related research, with respect to both climate mitigation and adaptation. These fields produce a multitude of complex, heterogeneous, geo-referenced climate related networks. Due to the size and the different properties of such networks, their investigation is not trivial. In the sense of the counterpart to sophisticated machine learning algorithms, visual analytics methods are crucial analyzing these networks visually, interactively keeping the climate researcher in the investigation loop. Existing visualization solutions can tackle the specifics of these networks only partially, in particular, they have problems with the size, geo-reference, their interlinkage, and the time-dependency of these kinds of complex networks. Filling this gap, we have developed a new visualization tool, which intensively uses the abilities of sophisticated graphic card processors to process large amounts of network data in a very fast and parallel manner. The abilities and flexibility of the proposed approach are illustrated for a classical climate teleconnection example and for a temperature-based infection disease network on flight routes.

I. INTRODUCTION

With increasing the computing power of high performance computing devices and an increasing number of measurement sensors and remote sensing abilities, the amount and size of produced datasets are ever more increasing. This is in particular true for the Earth system and climate data, where the exploitation and exploration of the gathered data become a major bottleneck.

An established solution addressing this challenge is the *complex network analysis* of geo-spatial climate data, exploring teleconnection and other kinds of interactions within the climate system.[1] These networks combine the ability to represent complex interrelationships in complex systems with statistical tools that can characterize the topology of the interrelationship pattern. Here, a complex network is a graph $G \in \{V, E\}$ with a set of nodes $V = \{1, \ldots, N\}$ and edges E. The network can be represented by the adjacency matrix A_{ij}, with $A_{ij} \neq 0$ if $\{i, j\} \in E$. The edges can have weights $A_{ij} \in \mathbb{R}$, then we call it a weighted network. In an unweighted network, the edges are represented by $A_{ij} = 1$.

This branch of climate research is producing multiple kinds of large geo-referenced networks, including temporal-evolving ensemble and cross-connected multi-layered networks. Two prominent examples are teleconnection networks of the global climate system,[2,3] which have as well been studied using interactive network visualization techniques for data investigation.[4] For example, Boers *et al.*[3] revealed the global coupling pattern of extreme-rainfall events and allowed us to identify the attribution of regional weather systems and upper-level Rossby waves on the rainfall pattern. In addition, networks from other fields are gaining increasing relevance for climate related studies, such as supply chain networks or flight networks.

Here lies the entry point for our research: the visual exploration and presentation of climate related networks (consisting of sets of pre-calculated geo-spatially embedded nodes and edges), using 2D, 2.5D, or 3D node-link diagrams. These networks and their visualization impose several challenges. In addition to their size—often resulting in cluttered images or 3D node-link diagrams—and diverse characteristics, climate researchers require multiple visualization tasks to be performed. These include *the overview of the spatial network structure*, *temporal evolution of the overall edge structure in combination with gridded climate data*, *subnetwork exploration in space, scale, and time*, down to the *exploration of the connectivity and the temporal behavior of individual network nodes*.

General purpose network visualization solutions tackle only parts of the solution space required and have some restrictions handling the networks at hand. To provide an example, the *Gephi* tool is strong in network layouts and network measure calculation; however, the options for visualizing networks with a given layout on a geo-physical base layer are restricted. The *Tulip* tool provides a multitude of network measures and visualization techniques as well with geo-information, but still misses a 3D option for multi-layered networks. Finally, d3js provides multiple options for network visualization including edge bundling in the geo-spatial reference, however, is suitable for smaller networks of several thousands of edges only.

Against this background, we decided to develop a dedicated tool providing solutions to the multitude of network data characteristics and tasks, based on a prototype originally developed for flight movement data, the Geo-Temporal eXplorer (GTX). This tool has been published at www.gtx-vis.org (including the source codes). It is based on a Graphics Processing Unit (GPU)-based visualization pipeline that systematically applies the following principle: all input data, i.e., the network data of nodes and edges, as well as all attributes and additional data, are uploaded directly onto the GPU in form of complex data, rather than geometry. It is only during rendering that the data are processed and the geometry is generated with respect to the visualization mode and settings currently selected by the user.

This approach has the following advantages:

- Data upload from the CPU to the GPU is required only once at the beginning. All data processing, filtering, and geometry generation are executed entirely on the GPU.
- An update of visualization settings by the user does not require to re-upload large parts of the data/geometry to the GPU, only the configuration settings need to be updated. This enables instant modifications of the visualization.
- Geometric representations can be altered entirely from one frame to the other, just by modifying the settings of the visualization pipeline.
- The complex data on the GPU can also be used to spawn processing and calculation processes on the GPU in parallel to the visualization process. For example, the adjacency information of nodes and edges can be easily processed in order to calculate the k-neighborhood. The results of these additional computations are in turn used as input data to the visualization process, e.g., to visualize the selected edges in the k-neighborhood.

As a disadvantage of a GPU-based implementation, we see slightly higher implementation efforts, having in mind the development of algorithms (such as cartographic projections) from the scratch in our case. The implementation challenges arise from the processing and storage of heterogeneous climate network data as direct input into the visualization pipeline, rather than from geometric primitives. In addition, the main memory of the GPU is the main limiting factor restricting the size of applicable networks.

At a glimpse, our tool contributes the following functionalities:

1. handling at a minimum 1×10^6 edges interactively,
2. browsing multiple time steps/ensemble members,
3. interactive highlighting of individual nodes and the related subnetworks (focus and context),
4. time charts for interactively selected nodes,
5. on-the-fly change of geo-projections, and
6. multiple network layers in 3D.

The high-performance implementations of these functionalities within GTX rely on GPU algorithms such as shader programs, GPGPU methods, and a GPU-based processing and visualization pipeline, which will be described and evaluated within this paper.

For this evaluation, we present two use cases both described in a high degree of detail for the interested physicist reader. The first use case illustrates how an ensemble of seven large networks with rising size and complexity (from 75 K to 526 K edges) can be visually explored at interactive frame rates, gaining new insights from former hair-ball representations. The second use case of temperature bridges in a flight network was selected to illustrate the flexibility of GTX, among others allowing to combine pre-rendered temperature choropleth maps with time-dependent daily sub-networks in several geo-projections. In addition, both use cases showcase GTX' interactive edge and node filtering as well as individual node selection functionalities enabling user-driven reductions of the network size and complexity without losing the context of the remaining network.

The paper is structured as follows. First, we provide a glimpse into the related work (Sec. II). In Sec. III, we describe the GTX tool and its components. Afterward, we describe their individual GPU implementations (Sec. IV). This is followed by two case studies (Secs. V and VI), and a discussion and a conclusion section (Secs. VII and VIII).

II. RELATED WORK

GPU-supported visualization allows rendering of complex scenes with increased rendering performance and decreased memory consumption. Simple geometries such as lines or splats in combination with GPU-based texturing and shading have been applied.[5,6] To enable users to explore geo-data at different levels of complexity, visual representations in 3D geo-virtual environments basing on GPU-based techniques (supporting focus+context, level-of-detail, and level-of-abstraction) have been developed.

The visualization of the geo-referenced network data has been reviewed under different viewing angles: from graph drawing,[7] information visualization,[8] cartography,[9] and with an application perspective.[10] Recently, Schöttler *et al.*[11] have provided a design space for visualizing and interacting with geo-spatiol networks. Several publications survey challenges and solutions related to large networks visualization.[12,13] Tools developed to interactively visualize large networks with more than 100 000 edges are Gephi (gephi.github.io) and Tulip (tulip.labri.fr). However, they have their limitations when it comes to the interactive handling and filtering of several million edges.

Filling this gap, several new approaches for graph drawing and rendering have been developed in recent years. Brinkmann *et al.*[14] proposed a GPU implementation of the ForceAtlas2 algorithm, which later has been extended for on multiple GPUs in a tiled display environment.[15] Linsenmaier[16] accelerated the Brinkmann *et al.* algorithm implementation[15] to a speedup of three to six times by using an optimized version of the Barnes Hut algorithm. In addition, network algorithms have been ported to the GPU. For instance, the GPU implementation of the KD-tree algorithm basing on a breadth-first search strategy introduced by Zhou *et al.*[17] allows a fast *k*-neighborhood calculation.

Beyond planar network representations, interactive 2.5D visualization techniques have been established using virtual globes. Alper *et al.*[18] propose a globe visualization "imprinting" the network into a deformed virtual globe. Alternatively, by mapping the length of an edge to the height of a 3D arch, short and long teleconnections can be visually distinguished easily.[19]

In this work, we address the challenge of temporal/evolving and ensembles of networks. Related visualization solutions include the use of animation, space–time cubes,[20] and temporal edge bundling.[21] A tailored solution for smaller and medium sized evolving networks is the Graph Stream library (graphstream-project.org).

Summarizing, existing visualization solutions have limitations with respect to the size of interactively explorable networks and the flexibility handling of the geographic/spatial context and, thus, are only partly applicable for large climatic networks.

III. REQUIREMENTS AND DESIGN RATIONALS

GTX has been designed as a tool for the interactive visualization of large geo-referenced network data. It supports interactive exploration and presentation tasks of two- and three-dimensional, time dependent, and multi-layer networks within their geographic contexts. In the following, the main design rationale of the GTX tool is described:

Large datasets: Geographic networks are often large in size. While the number of nodes usually ranges between 1000 and 300 000, the number of edges can easily exceed tens of millions of edges or even more for dense networks. Time-dependent, multi-layer, and multi-scale networks increase the size even further, as they in fact add one full network for each layer or point in time, respectively. Therefore, large networks of sizes up to at least 10×10^6 nodes and edges must be supported, regarding processing and visualization at interactive frame rates.

Geographic embedding: In contrast to general graphs or networks, the positions of nodes in geographic networks are not arbitrary but carry a geographic meaning and are very important for the interpretation of the underlying geo-physical processes and, therefore, need to be preserved. In order to set geographic networks into context, a visualization has to include not only the network itself, but also its geographic surrounding, using a visual representation such as a geographic map or a virtual globe. As different types of networks require different visualizations for their interpretation (e.g., global vs. regional, 2D vs 3D), several diverse geographic projections must be supported, and it should be possible to select and switch between different geographic environments interactively.

Interactive filtering: To explore large networks interactively, filtering needs to be applied. In particular, nodes and edges need to be filtered spatially (based on their geographic position), temporally (based on the time-attribute in temporal networks), and attribute-based (based on the value of node and edge attributes). As filtering is part of the user's exploratory process (e.g., for formulating and testing hypotheses of climatic processes), instant and interactive filtering is required.

Interactive mapping: For the analysis of node and edge attributes, a dynamic mapping process is needed. For example, attribute values may be mapped to the height, size, or color of nodes and edges. Also, different mapping configurations, such as color gradients, need to be applied to visually distinguish between various subgroups of the network, e.g., different layers of height may use different color mappings. Further, interaction states, such as selection and filtering, must be communicated using different visual mappings.

Focus & context: Setting the focus on individual nodes and depicting the related sub-networks in combination with keeping information of the context enables climate researchers to study the embedding of certain nodes within rest of the network. In particular, this is relevant for analyzing network hubs.

Evolving temporal networks: To analyze temporal evolving networks, their specific semantics need to be taken into account. In time-dependent network data, not only attribute values of nodes and edges can change over time, but nodes and edges can also be added or removed from the networks. While it can sometimes be sufficient to treat a network at one given point in time separately, the union of nodes and edges over time also needs to be preserved, as to, e.g., analyze the development of attribute values over time, or to monitor changes in network topology over time.

3D multi-layered networks: Multi-layered or coupled networks are composite networks that consist of several layers, corresponding for example with data from different atmospheric levels. Therefore, such networks often contain a third dimension, signifying the height of a node's position. Each layer can either be interpreted as its own network, which may be analyzed separately and may also contain attributes specific only to that layer. But also interconnections between different layers exist, which can be of high importance for the analysis and understanding of the entirety of the coupled network.

IV. FRAMEWORK IMPLEMENTATION OVERVIEW AND DETAILS ON GPU-BASED MAPPING AND RENDERING

The requirements stated in Sec. III pose a lot of challenges for an interactive system. Not only do large datasets need to be processed and rendered at interactive frame rates, but also the visual representations of the data need to be updated interactively, in order to enable dynamic mapping and exploration by the users. Additional analytic operations, such as calculating k-neighborhoods, have to be performed interactively as well. Therefore, GTX applies a set of GPU-based techniques that enable not only fast rendering of large datasets but also flexible filtering, mapping, and processing of data on the GPU. As a result, a data-driven visualization pipeline has been implemented, which combines data processing and visualization into a single, GPU-based rendering process.

Figure 1 shows the principle of this GPU-based visualization pipeline. First, the network data are loaded from file and uploaded to the GPU, represented as an attributed vertex cloud. This GPU representation contains the original data of the network, i.e., the position of nodes in latitude and longitude, as well as the values of all node and edge attributes. During rendering, the dataset is processed, filtered according to the current spatial, temporal, and attribute-based filtering settings, then the geometric representations for all unfiltered nodes and edges are generated on the fly, taking into account the current mapping options provided by the user. The generated geometry then, finally, is rendered and rasterized.

Since all operations take place on the GPU during the rendering process, the results of user interaction are instantly visible. For example, filtering and mapping options are represented on the GPU as uniform shader variables, which are only a few bytes in size. User interaction that influences filtering and mapping, therefore, only requires these configuration values to be updated, which is accordingly fast. Afterward, a rendering update is triggered, which causes the updated visualization to be produced in the next frame. Therefore, even seemingly large modifications to the visualization settings, such as switching between geographic projections, which cause all nodes and edges to be at completely different locations afterward, can be performed instantly, as no re-processing of data and geometry generation on the CPU, and no uploading of data to the GPU are required in the process.

GTX has been implemented using OpenGL 3.2 API and shader programs. The individual components are described in Subsections IV A–IV H.

A. GPU representation of network data

GTX provides readers for dot, graphml, and xml file formats. Multi-variate attributes on nodes and edges can be imported. In addition, for certain network classes, multiple networks can be imported simultaneously, for example, representing different network layers or network time series. This data are first loaded into a CPU data structure.

Rather than generating and uploading geometric representations such as textured triangle meshes onto the GPU, as is done by most visualization systems, the actual multi-variate input data are uploaded to the GPU, represented in the form of vertex arrays and

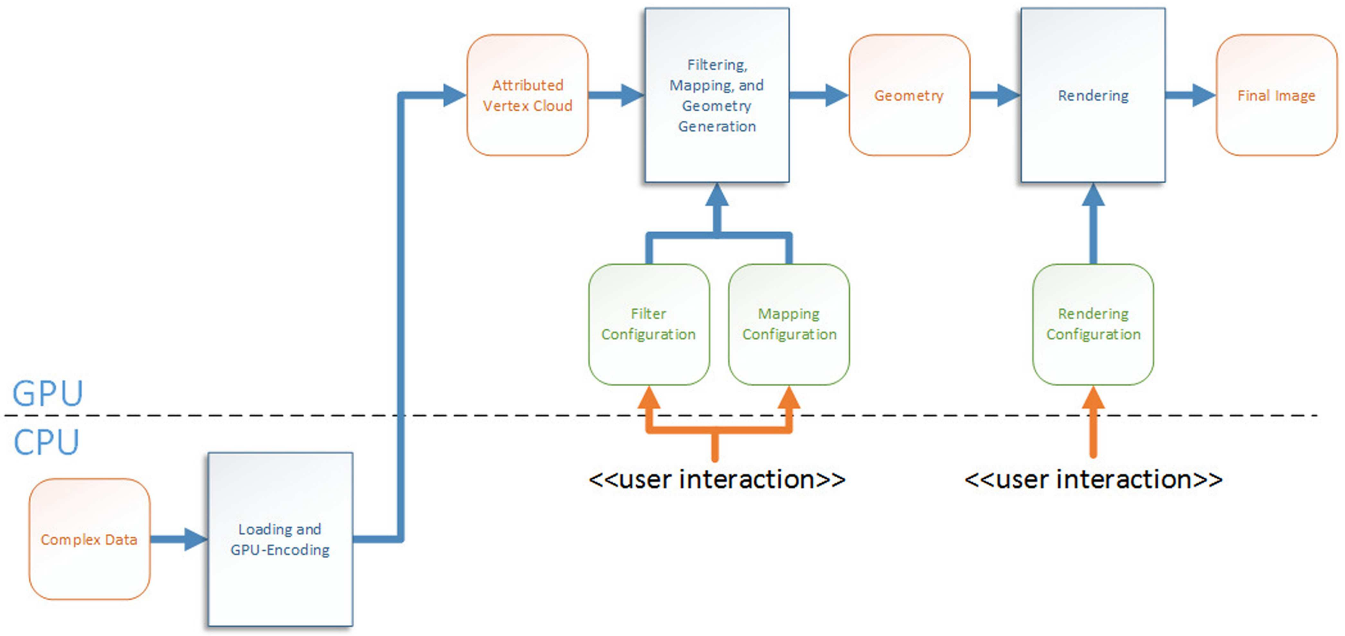


FIG. 1. Principle of the GPU-based visualization pipeline.

buffer textures. These data are processed and then transformed into actual geometry only during the rendering of a single frame.

Node geometry: Nodes are stored in a single vertex array. Each data item contains the node ID, the position in latitude and longitude, and a time stamp for temporal networks. The ID of a node also represents an index into the attribute buffer, at which the attribute values for that node can be retrieved.

Edge geometry: Edges are stored in another vertex array. Each item contains the edge ID and the node IDs for the outgoing and incoming nodes. As analog to nodes, the edge ID also serves as an index into the attribute buffer.

Attribute data: The values of node and edge attributes are stored in a single floating-point texture, using the texture type *GL_BUFFER_TEXTURE*, which supports storing large amounts of linear floating-point data. In this large array, the values of all attributes of all nodes and edges are stored in a linear manner, so given the ID of the node or edge, and the ID of the respective attribute, the attribute value can be retrieved in constant time or O(1).

As described above, this representation of the network data does not represent a specific geometry, such as a node-link diagram of the network using spheres and lines, but it contains the network data itself. It can, therefore, be used not only to produce a specific visualization of the data, but also to process the data in terms of GPGPU operations. This can be used for example, to perform analytic computations on the network data or to execute graph algorithms, which are needed as part of the visualization process.

B. Visualization state

The visualization state contains the current options a user has selected, including the state of interaction with the visualization, such as the currently selected or hovered items. The state is communicated to the GPU visualization pipeline by setting shader uniform variables and, in some cases, using additional buffer textures to represent larger data.

Geographic projection: This option defines the kind of geographic mapping that is currently selected. It determines the kind of geometry that is rendered for the geographic environment and also which shader function is used to transform from geographic coordinates (latitude and longitude) into the virtual 3D space when rendering the actual network geometry.

Attribute mappings: This includes the attribute mappings for node color, size, and height, as well as edge color and width. For each of these, the index of the specific attribute is stored.

Filter values: For each attribute, a pair of values is stored, representing the minimum and maximum values of that attribute. These filter data are stored in a texture, which can be used to look up the minimum and maximum values for a given attribute ID. Also, for temporal networks, the current minimum and maximum time stamps are stored.

Selection-buffer: The selection-buffer contains the list of IDs of all currently selected nodes. It has a fixed sized, so only a certain number of nodes can be selected at one time. It is represented on the GPU using a uniform buffer object.

Hovered node: In addition to the selected nodes list, this stores the ID of the node currently hovered by the mouse.

K-neighborhood: This represents the current k, which defines the size of the selected neighborhood.

Selection-texture: The selection-texture is a buffer texture, which for each individual node contains the information if it is currently selected or within the current k-neighborhood of any of the selected nodes.

C. Virtual environment rendering

At first, the geometry for the virtual environment into which the network diagram will be embedded is rendered onto the screen. This can be either a 3D virtual globe or a 2D map. Therefore, depending on the current visualization settings, either a textured 3D sphere or a texture 2D rectangle positioned in 3D space will be rendered.

In the case of the 3D globe, the UV texture mapping is determined by a spherical mapping of a map texture onto the globe. In the 2D case, however, several geographic projections are supported. This is implemented in terms of shader programs: For each pixel on the 2D plane, the reverse projection function is applied to determine the coordinates for the texture lookup. This enables the geographic projection to be switched on the fly: when a different projection function is selected, only the projection function in the shader pipeline needs to be changed for the next frame.

Also, depending on the currently selected virtual environment and 2D projection, a shader program is selected that will later be used to transform the geo-position of nodes and edges into the 3D positions in the virtual environment.

D. Rendering of network data

In the next step, the actual geographic network visualization is created by generating a three dimensional node-link diagram of the network. This is initiated by rendering the attributed vertex clouds, representing the network's nodes and edges. This causes each item to be processed in parallel on the GPU, and for each item, the entire visualization process—filtering, mapping, and rendering—is performed. During this process, the actual geometry is generated from the input data, taking into account the current visualization state, such as the geographic projection and attribute mapping, then the generated geometry is rendered to the screen.

The processing of an attributed vertex cloud is triggered with a single draw call, which draws the entire vertex array in a single step (*glDrawArrays*). This enables a very efficient processing of all data items, as the amount of draw calls is essential for the performance of an interactive graphics application. The processing and rendering is implemented in terms of shader programs:

Data processing and filtering: The vertex shader processes the data item and fetches the data attributes for the node or edge, which are stored in the attribute-texture. These attribute data are compared with the current spatial, temporal, and attributed-based filter values, and data that is filtered out will not be processed any further, so no geometry is generated for them.

Geometry generation: In the geometry shader, the actual renderable geometry for each data item, which have not been filtered already is generated. In this step, the attribute data, which has been fetched before, are applied to the visual attributes of the geometry, such as the color or size of a node. Also, the actual 3D positions for each data item is determined by applying the projection from geographic coordinates into the 3D space, according to the currently selected geographic projection.

Node geometry: The nodes of the network are represented by spheres, which are rendered using splats to improve rendering performance. Splat rendering represents the 3D geometry of an object by sampling the surface by points that come with a surface direction vector and renders them by texture shaders. "The idea is to approximate local regions of the surface by planar ellipses in object space and then render the surface by accumulating and blending these ellipses in image space."[22] This enables large amounts of nodes to be rendered in real-time with low memory consumption for the generated geometry. The spheres provide radius and color as visual variables onto which node attributes can be mapped.

Edge geometry: Edges are represented by lines between the connected nodes. They can be either straight 2D lines or 3D arcs. Line color, line width, and arc height are available as visual variables for the mapping of edge attributes.

Rasterization: Finally, the generated geometry is rasterized onto the screen. In the fragment shader, texture mapping and color lookup are applied to determine the final color for each resulting pixel.

E. Time series view

For the visualization of temporal networks, a planar time series view has been implemented, depicting the temporal evolution of user-selected node attributes (see Fig. 6). It displays the value of a single attribute of a selected node over time, rendered as an UI element on top of the network visualization. To generate this visualization in real-time, the attributed vertex cloud of the nodes is processed a second time, using a different set of shader programs.

Data processing and filtering: In temporal networks, the vertex cloud contains each node several times, once for each recorded time stamp. Each item has an individual item ID, which is used to fetch the attribute data for this particular instance, but also contains a shared node ID to identify which items represent the same node, but at different time stamps. In the vertex shader, this node ID is determined and compared to that of the currently selected node. Only items which represent the selected node are processed further, otherwise the processing is omitted at this point.

Geometry generation: For the selected node, the time stamps of the current data point are read and mapped to the X-axis of the visualization. The value of the selected attribute for that node is fetched from the attribute data texture and mapped to the Y-axis. From these data, a line geometry is generated in the geometry shader, which is then rasterized onto the screen to create the actual value plot.

F. Selection and picking

In addition to the resulting color image of the visualization, an ID map is generated during the rendering process, which contains the ID of the displayed item—node or edge—for each rendered pixel. When the mouse is moved over the visualization, the ID of the currently hovered node is determined by looking up the pixel in this ID map. This information is then propagated to the rendering state to visually highlight this node.

By pressing the mouse button over the object, the user can select and deselect the current node. This information is stored in the selection-buffer on the GPU and used both for highlighting the selected nodes and for calculating the neighborhood of the currently selected nodes.

During the rendering process, the currently hovered node is visually highlighted, as are all selected nodes. In addition, the nodes and edges within the k-neighborhood (see Sec. G), which are considered to be in the focus of the visualization, are displayed with a prominent rendering style, while the other nodes and edge (the context) are visually faded out.

G. Neighborhood calculation

The k-neighborhood of the currently selected nodes is also calculated on the GPU [K-nearest neighbors (kNNs) is a supervised classification algorithm that determines the classification of a point X by using the classification of the K-nearest points of X]. This process is a shader-based implementation of Bellman Ford's shortest-path algorithm, modified only by allowing multiple start vertices to exist. It is triggered whenever the selection of nodes has been modified by the user. The neighborhood information is stored in the selection-texture: for each node, it stores the number of steps that is needed to get from one of the selected node to this node.

The process starts with the clearing of the selection-texture: each element is set to a value that indicates that the node is unreachable (-1). Then, the value for each selected node is set to zero (0), indicating that the node is reachable within zero steps. The neighborhood is now calculated using a ping-pong strategy: a second texture has been created with the same size as the selection-texture. One of the textures is used as input, the other is used at the output texture. After each step, the textures are switched.

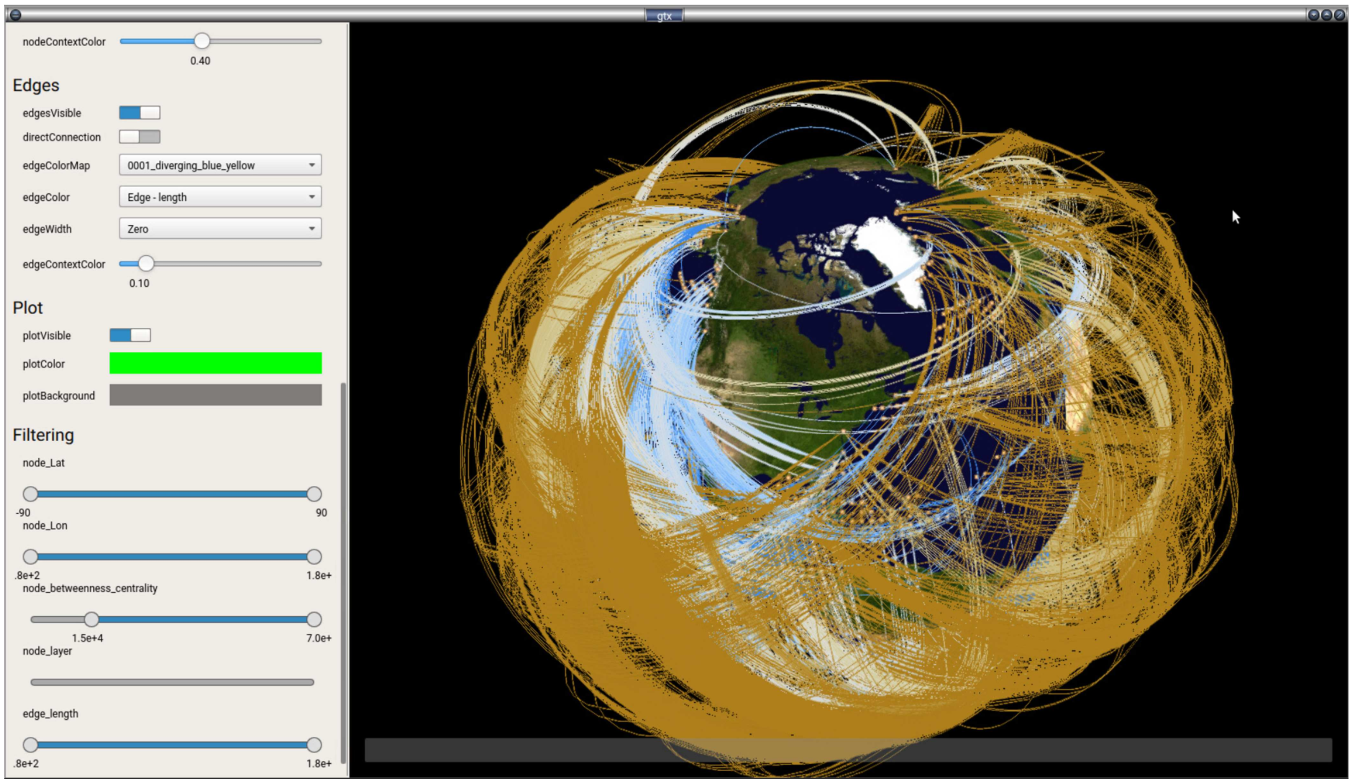
In each iteration, the attributed vertex cloud containing the edges of the network is rendered. In the shader program, the IDs of the source and destination nodes are retrieved, and the current value for the destination node is fetched from the input texture. If that value plus one is smaller than the value of the destination node, the new value is written into the output texture at the position of the destination node. Otherwise, the fragment is discarded. The process is repeated exactly k times for a selected k-neighborhood, after which the neighborhood has been fully calculated.

H. User interface and animation

To enable user to interact with the system, we developed a QT5-based graphical user interface (see Fig. 2). This interface consists of a parameter dialog for dataset selection and import, and controls for all parameters of the filtering and rendering process, including the interactive node and edge filters as well as size and color definitions.

In addition, a JavaScript-based script interface was developed. It allows to import time-dependent or ensemble network datasets in combination with a set of pre-calculated earth images (e.g., for pre-rendered earth-maps in Figs. 4–6) and predefined camera path definitions for animated videos productions.

Individual base maps in combination with sub-networks defined for individual time steps or ensemble members can be



nodeContextColor 0.40

Edges

edgesVisible

directConnection

edgeColorMap 0001_diverging_blue_yellow

edgeColor Edge - length

edgeWidth Zero

edgeContextColor 0.10

Plot

plotVisible

plotColor

plotBackground

Filtering

node_Lat -90 90

node_Lon .8e+2 1.8e+

node_betweenness_centrality 1.5e+4 7.0e+

node_layer

edge_length .8e+2 1.8e+

FIG. 2. The graphical user interface of the GTX Explorer; left: edge rendering as well as node and edge filter configuration options; right: a climate interaction network (from Use Case 1 in Sec. V), with 526 000 edges interactively filtered by node betweeness-centrality and color-coded by edge length.

animated using either a time slider or a forward+backward key definition applying the scripting language.

V. USE CASE 1: VISUAL EXPLORATION OF LARGE MULTI-TIME SCALE CLIMATE NETWORKS

Using the GPU-based visualization pipeline described in this work, an interactive visualization method for large 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. Due to the embedding of climate networks into the geographic background, node positions need to be fixed according to their geo-position. Position-changing layout 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. 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 and finding spatially linked climate relevant regions (e.g., for detecting climate teleconnections). On the other hand, the 3D visualization can also make the perception of the visualized data more difficult. Therefore, additional techniques such as filtering and edge bundling must be applied to improve the visual perception of the network structure.

A. Background

The climate system can be considered as consisting of interacting subparts, whose interactions determine and control the properties of the climate system. The method of complex networks is a powerful tool to represent and investigate the topology and mechanisms of such systems that consist of many interacting parts. For this purpose, spatially embedded (station-based or gridded) climate data are transformed to a complex network representation on the basis of statistical interrelations (e.g., the Pearson correlation between the time series of the considered stations or grid points). Such climate networks can be used to understand the spatial connections of temporal interrelations in climate related processes such as rainfall propagation or sea surface temperature. These studies have used network measures such as degree, clustering coefficient, and betweenness centrality to quantify the spatial connections in the climate system. This approach has been used to uncover climate teleconnections and to develop prediction schemes.[1,3,23]

Climatic processes are generally scale-dependent processes and exhibit certain characteristics that are related to other scales. The scale relationship can be temporal, spatial, or both. Understanding how these scales are connected and how the geophysical processes evolve with the scale is vital in many applications such as disaggregation, downscaling, upscaling of data, and identification of scale specific and emergent processes. Many approaches such as fractal analysis and self-similar behavior using moments provide insight into the scaling behavior of the geophysical processes. However, these methods are not capable of capturing the spatial connections of the given climatic processes in the neighborhood. A combination of wavelet transform and complex network approach can help to investigate the spatiotemporal relationships on different scales. This approach is robust[1] and can provide detailed insights into the process which remain generally obscure at single scale investigations. For instance, several studies have used wavelet transform to decompose surface air temperature and global sea surface temperature data and applied climate networks at corresponding timescales to identify different hubs at different scales responsible for known atmospheric circulation phenomena or to uncover long-range connections within the climate system (teleconnections) that were beyond the hitherto known.

B. Data characteristics

The climate network studied here is based on the global monthly sea surface temperature (SST) data provided by US National Oceanic and Atmospheric Administration—Earth System Research Laboratory's Physical Sciences Division (NOAA/OAR/ESRL PSD). The data are freely available at https://psl.noaa.gov/data/gridded/index.html. The data (ERSST V3b) have a spatial resolution of $2.0^\circ \times 2.0^\circ$ and are given for the time period 1979–2015. As a preprocessing step of the SST data considered here, we have removed 1056 grid points out of total 10 512, with missing values or gaps in the SST data, hence, in total, 9456 grid points considered in this study. Further, we calculate anomaly series by subtracting the climatological mean for each month of the time series, which significantly reduces temporal auto-correlation in the timeseries.

C. Visualization examples

To construct the climate networks, each SST grid cell is considered as a node and edges are created between all pairs of nodes based on statistical relationship. The similarity measure used is the wavelet multiscale correlation (WMC).[1] For the WMC, the data are first decomposed at different timescales and then the Pearson correlation between all pairs of nodes is calculated at corresponding timescale. Finally, significance based pruning is applied to retain only highly correlated edges in the network.

The network is constructed by applying a threshold to the WMC values. A number of criteria have been used for thresholding, such as a fixed amount of link density or fixed thresholds. Here, we consider a 5% link density since it is a well accepted criteria globally for network construction. However, here we combine it with multiple testing attempts to avoid false links by controlling the type I error or adjusting p-values to give only significant links. Betweenness centrality (BC) is calculated for the network nodes, which is a measure of control that a particular node exerts over the interaction between the remaining nodes. In simple words, BC describes the ability of nodes to control the information flow in networks. To calculate betweenness centrality, we consider every pair of nodes and count how many times a third node can interrupt the shortest paths between the selected node pair.

D. Results

The network visualization of the original SST data (all scales) reveals short and long range connections between many regions of the Earth [Fig. 3(a)]. For example, we can identify short range links within the Indian Ocean and the tropical Pacific, which could be related to the Indian Ocean Dipole or the El Niño/ Southern Oscillation (ENSO). Long range connections appear between the western Indian Ocean and the Atlantic and the tropical Pacific with the Mediterranean Sea. However, we cannot distinguish at which timescales these connections appear. To get a clearer picture, we look at the scale-dependent networks.

Considering short timescales, we can attribute mainly the short links to these timescales [Figs. 3(b)–3(d)]. At longer timescales, the long range connections appear. At timescales of roughly a year, mainly the tropical oceans interact with each other [Fig. 3(e)], whereas at approximately 2 years, the global impact of the ENSO region is obtained [Fig. 3(f)].

The GTX tool allows an easy visualization and comparison of the connections at different scales by handling the complexity of the networks data, interactive filtering of most important or selected edges, and attributing the interrelationships at certain scales to selected regions.

VI. USE CASE 2: VISUAL EXPLORATION OF TEMPORAL EVOLVING CLIMATE BRIDGES IN WORLD AIR-TRAFFIC NETWORKS

The world air-traffic network connects different regions to each other. Within only one day, a large number of people are able to travel around the globe across climate regimes, e.g., departing under cool and arriving under hot weather conditions and vice versa or similar conditions on two connected airports. These two components, climate and mobility, can affect the fast spread of temperature related infectious diseases and pathogens.[24] Against this background and to identify critical constellations in the whole climate-mobility-system (CMS), the idea of this use case is to analyze the "climatic" characteristics of airport nodes. Associated with this, temporary flight connections or subnetworks can be identified in which the prevailing temperature conditions favor the possible survival conditions of pathogens.

A. Background

The impacts of climate change onto the spreading of infectional diseases is co-determined by the temporal developments of climatic factors:[25]

- The transmission of mosquito-borne infectious diseases is affected by factors including temperature, humidity, and precipitation. For malaria, the number of months suitable for transmission of *Plasmodium falciparum* and *Plasmodium vivax* malaria parasites is calculated on the basis of temperature, precipitation, and humidity.

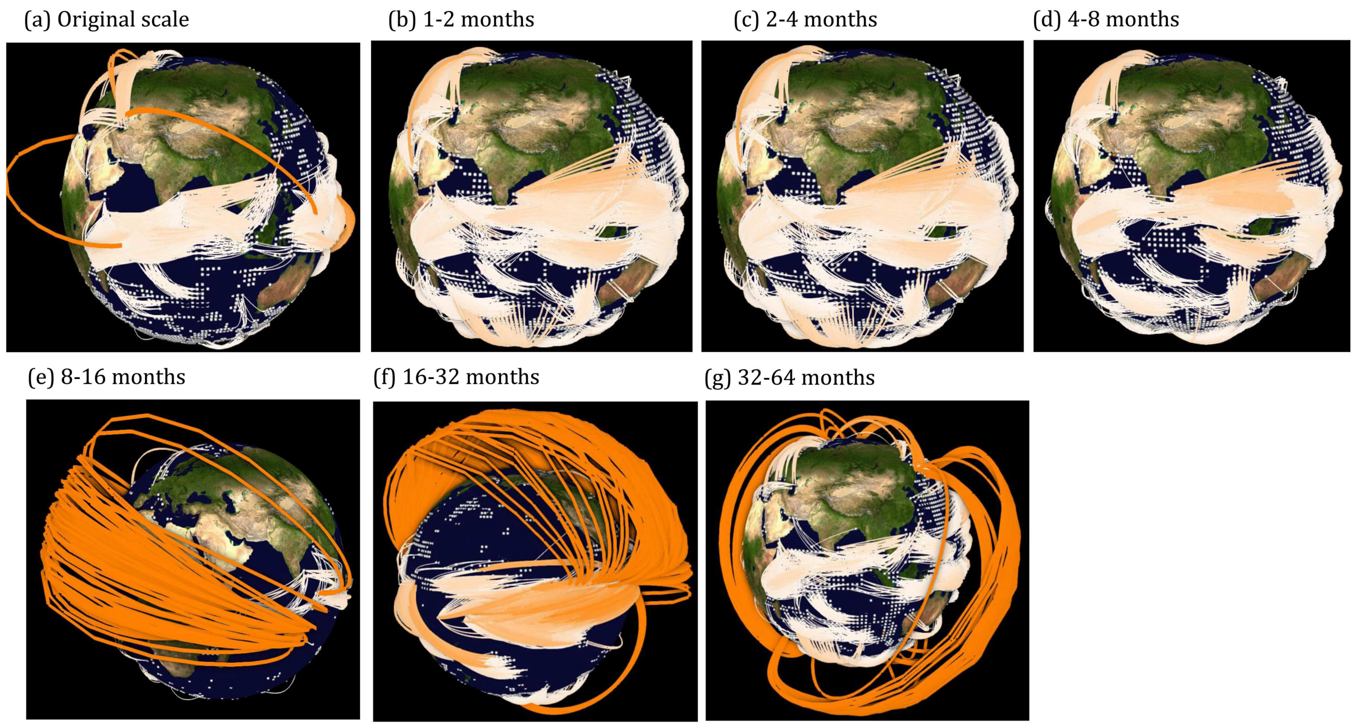


FIG. 3. Spherical three-dimensional globe representation of the short and long-range teleconnections at different timescales in the sea surface temperature network (originally published in Ref. 1). Only nodes beyond a pre-selected betweenness centrality (BC) value are plotted. For instance, in subplot (a)–(d),(e) and (f), and (g) nodes are plotted for BC values greater than 90 K, 57 K and 38 K respectively. Edge color represents the geographical lengths.

- Vibrio species cause a range of human infections, including gastroenteritis, wound infections, septicemia, and cholera. These bacteria are found in brackish marine waters and cases of infections are influenced by sea surface salinity, sea surface temperature, and chlorophyll.

The actual dissemination of pathogens in the air transportation system is hard to track based on the data. Expert interviews and laboratory experiments[26] at idealized conditions help prioritize potential influencing factors. Based on simplistic transmission models and using predefined transmission rates, which are associated with air temperature and humidity, outbreak situations can be simulated.

How quickly an outbreak spreads and reaches out to other regions depends on the weather conditions along the flow of passengers. In a previous work, we have modeled the sensitivity of the spread of flu infections in aviation as a function of vapor pressure.[27] The characteristics of the flight network as well as the location and season of the outbreak largely determine the speed of a pandemic.

Following this line of argumentation, in this use case, we investigate the properties of so-called "climate bridges" as an indicator for infection spread risks within the global air transportation network by interactively exploring the properties of temporally evolving network conditions.

B. Data characteristics

In order to assess this possible influencing factor, we combine $0.5^\circ \times 0.5^\circ$ gridded daily maximum temperature data over the continents (W5E5 dataset: https://doi.org/10.5880/PIK.2019.023) and open flight data (https://openflights.org/data.html). First, we define a reduced air-traffic network focusing on the 99 largest and most important airports in the world. This also includes approximately 3500 daily flight connections, which are considered constant over time. Second, we define the climate location factors at every airport by selecting the nearest grid point in the temperature dataset. For every day and flight connection, the temperature conditions on the departure and destination airport are assessed.

Now, as a basis for climate bridge definition for each airport, a weight can be defined based on its temperature conditions as

$$\omega(T_i) = \frac{1}{1 + e^{-0.15\,(T_i - 27^\circ\mathrm{C})}}. \tag{1}$$

Here, -0.15 and 27°C are constants, which determine the strength and the position of the maximal temperature gradient. In Northern Hemisphere mid-latitudes, this threshold is a precursor of heat stress with impact on other human-health related aspects. Now, based on the per airport weights $\omega(T_i)$, the strength of a connected climate bridge ω_{ij} (network edge) can be defined by the medium weight of the two airports i and j,

$$\omega_{ij}(T_i, T_j) = 0.5\ \left(\omega(T_i) + \omega(T_j)\right). \tag{2}$$

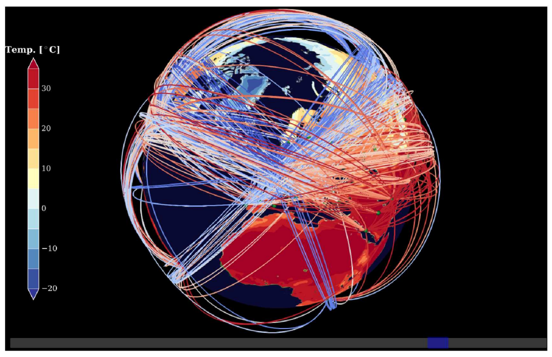
Based on this climate bridge definition, we have calculated node based network measures. The most relevant one in a weighted network is the degree centrality. This measure is the sum over all weighted edges (climate bridges) per airport. Then, the resulting patterns are visualized in a network view at the nodes together with the climate bridge strength on the edges and temperature on heat maps.

C. Visualization examples

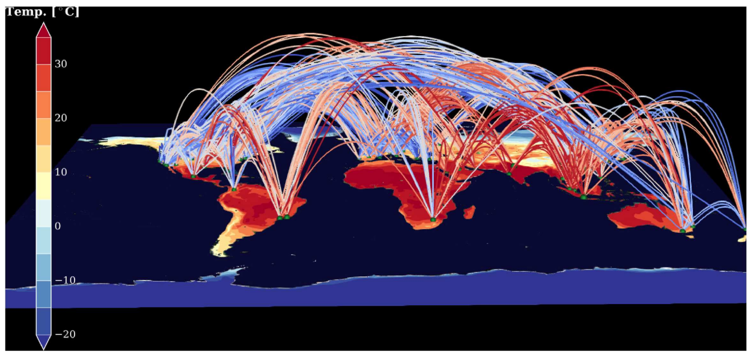
The visualizations in Fig. 4 illustrate the full network of flight connections in three different projections. Edges are color coded by their climate bridge strength $\omega_{ij}(T_i, T_j)$. A 3D spherical projection [Fig. 4(a)] best illustrates the spatial distances, however, major parts of the network are hidden. The Mercator projection [Fig. 4(b)], on the other hand, produces much clutter, in particular, by many overlapping long connections and by wrongly representing flight connections of airports layouted by the projection one to the left and the other to the right border [in Fig. 4(b): routes over the Pacific ocean]. Avoiding some of the shortcomings of Figs. 4(a) and 4(b), alternative projections have been integrated into GTX such as a 360° projection [Fig. 4(c)]. Here, with the North Pole in the center, based on the interactivity of the view, all relevant features of the network can be distinguished.

After selecting a suitable projection, the user can select individual airports and their connected networks and inter-compare the strength of climate bridges against the temperature conditions in different seasons (see Fig. 5). For the two selected airports, Frankfurt and Bangkok, Frankfurt shows the highest network activation under Northern Hemispheric summer conditions. Consequently, the degree centrality of Frankfurt is higher in summer than in winter. This also changes the airport ranking across seasons taking temperature-related flight routes into account.

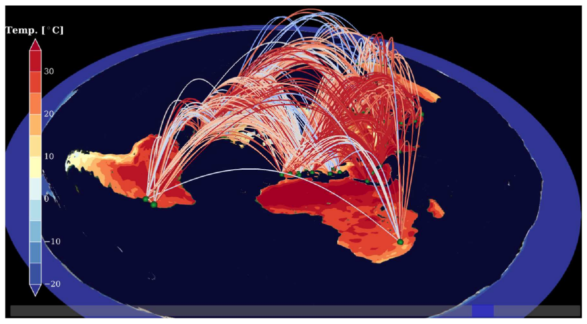
As a third example, Fig. 6 represents the interactive selection of two airports, focusing on direct connections (colored edges) and their context (semi-transparent gray edges). In addition, the annual cycle (June 2002 to August 2003) of the degree centrality is represented in the time series view (green color) to allow users to select



(a) 3D spherical projection



(b) Mercator projection



(c) 360° projection

FIG. 4. Representation of the full flight network in 2003 climatic conditions, illustrating varying occlusion levels in three implemented projections. (a) 3D spherical projection, (b) Mercator projection, and (c) 360° projection.

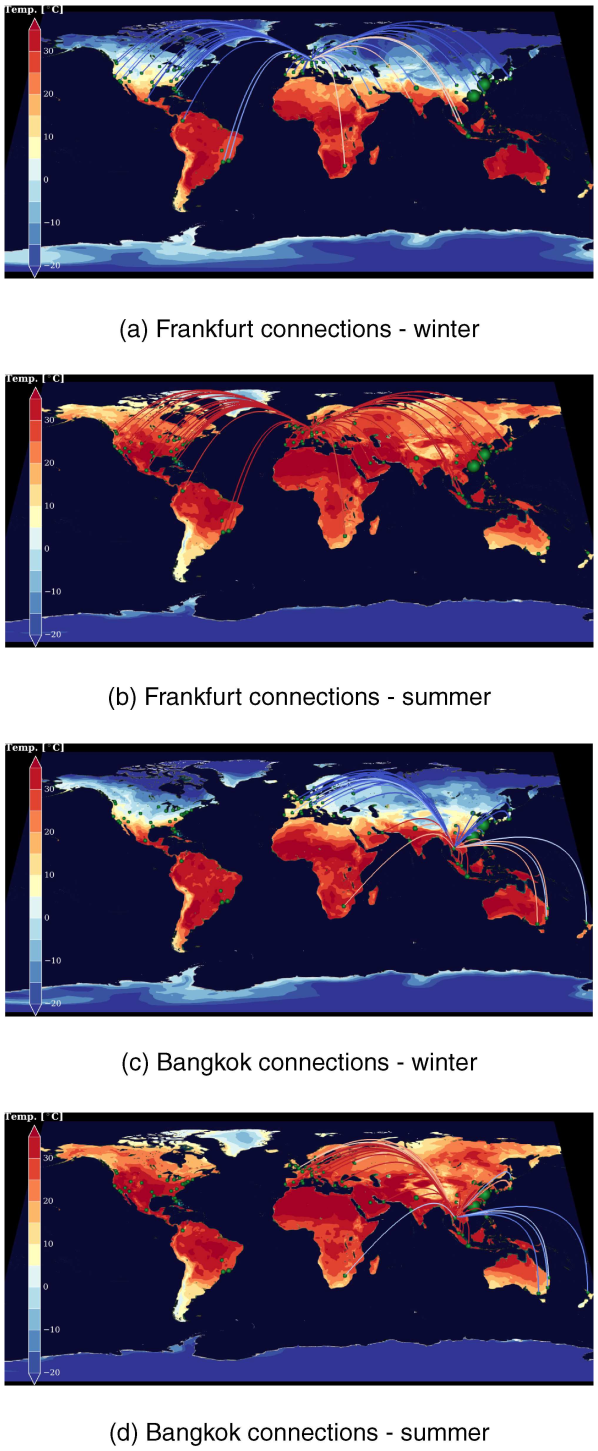

Temp. [°C]
30
20
10
0
−10
−20
(a) Frankfurt connections - winter
(b) Frankfurt connections - summer
(c) Bangkok connections - winter
(d) Bangkok connections - summer

FIG. 5. Winter and summer connection network of two airports (Mercator projection). (a) Frankfurt connections—winter, (b) Frankfurt connections—summer, (c) Bangkok connections—winter, and (d) Bangkok connections—summer.

time periods of interest. With the slider at the bottom of this figure, all possible conditions and large-scale temperature patterns can be set in relation to the selected airport and corresponding climate bridges.

D. Results

The presented climate bridges example—indicating a possible risk factor for spreading of infectional diseases—illustrates the potential of the GTX tool to analyze time-dependent features of air-traffic networks. The definition of climate bridges as used here to weight individual flight connections by climate factors can be understood as a transmission rate in order to simulate outbreaks as done by Refs. 27 and 28 and analyzing the speed sensitivity. The tool is highly scalable to further increase the number of flight connections beyond the presented example sizes. The combination of several visual attributes in the network view (node color and size, edge color and transparency) in combination with a color coded base map (with several projections) and an interactive time series view provides the required flexibility for this use case. Still, for longer time series, an improved network data handling and more sophisticated time series visualization methods would be desirable.

VII. DISCUSSION

Our approach implemented in the GTX explorer creates a highly interactive visualization system, as update costs to modify the visualization are nearly zero. Even large changes to the visualization, such as switching between 2D or 3D, selecting a different geographical projection, or even switching between a time series diagram and a node-link 3D view, can be achieved between one frame and the other, without detectable delay for the user.

The GTX explorer is running on two parallel nodes on a SUSE Linux Enterprise Server 12 system with a NVIDIA Tesla K40c graphic cards with 12 GB memory size. It is accessible using VirtualGL on an HPC system for more than 200 climate scientists. We tested its performance with several of the data subsets of use case 1 (see Table I). For the largest one with 9456 nodes and 526 000 edges with a 1300×1050 resolution display in a setting where all nodes and edges are not filtered and visible, the system still allows fluent interaction during zooming, panning, and earth rotation with a minimum of 17 MPixel/s and 14 frames/s. Up to a million edges can be loaded and concurrently processed in this hardware setting, limited only by the graphic card main memory.

However, the data processing on the GPU has as well a disadvantage: as the data are processed just on the GPU, it cannot be pre-filtered on the CPU to reduce the amount of data. For very large datasets, a limit to this approach, therefore, lies in the amount of data that can be uploaded to the GPU in terms of memory. Also, since processing and filtering takes place on the GPU, and each user interaction can completely change which items are visualized in which way, every data item has to be processed each time, even if it is then filtered out. This may impact the rendering performance on very large datasets. Also, since the generated geometry can also change drastically from frame to frame, the geometry is constantly regenerated, also impacting the rendering

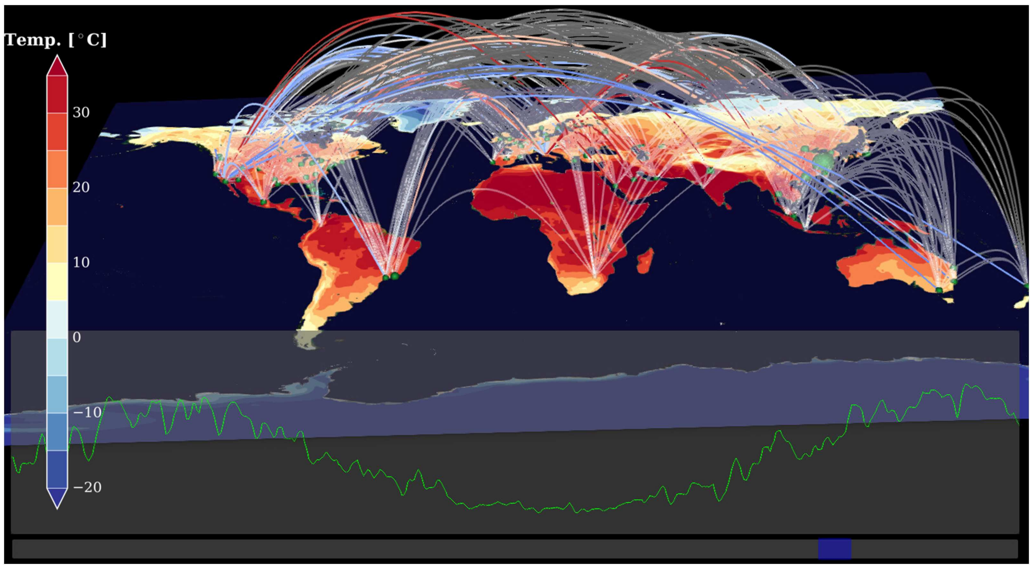


FIG. 6. The full flight network (context) and highlighted connections of Frankfurt and Los Angeles (focus); interactively selected time frame (blue); the line chart represents betweenness the daily development of the centrality of Frankfurt for two concurrent years.

performance. These disadvantages could be overcome by regenerating the geometry only after visualization modes have been modified (e.g., transform-feedback or similar GPU approaches). Finally, all processing and analysis functions on the data need to be implemented on the GPU as shader programs. For complex analytic algorithms, this may not always be feasible.

We are very aware that CUDA (Compute Unified Device Architecture) has become the state of the art over GPGPU implementations for network calculations such as k-neighborhood. Using CUDA for that would of course be possible, but is not straightforward to implement in conjunction with OpenGL. For that reason, we think it still makes sense in a visualization-focused application to implement everything with shaders (vertex, fragment, geometry, compute shaders), which have exactly the same capabilities as CUDA but are integrated directly into the visualization pipeline. We acknowledge that this comes with some implementation challenges but is possible for the skilled programmer.

Nevertheless, the tool allows a simple handling of multi-dimensional complex network data (temporally evolving, many node attributes, different scales), becoming more important in the study of climate related observation and model data.

TABLE I. Performance measurement of the interaction with four representative layers of use case 1.

Number edges	Total throughput [MPixel/s]	Frame rate [f/s]
75.000	50	41
171.000	41	34
312.000	25	21
526.000	17	14

VIII. CONCLUSION

This paper introduces the interactive network exploration tool GTX, which has been designed for the visualization of large geo-referenced networks. It contributes a GPU-based implementation that enables the interactive handling of more than 10^7 edges by executing all parts of the visualization pipeline, i.e., filtering, processing, mapping, and geometry generation, during the rendering process. This enables the implementation of highly interactive, use-case specific visualization components, such as spatial and temporal filtering and exploration, selection and highlighting, and temporal focus+context for temporal evolving networks. GTX enables users to choose an appropriate geo-projection just-in-time (e.g., to reduce clutter induced by standard layouts) and provides additional components, such as *k*-neighborhood computation, in real-time. Two examples from the climate science background have illustrated the applicability of the proposed tool.

The pre-processing of climate network measures in a parallel, very flexible python environment controlled by climate scientists is a very well-established workflow before entering the visualization session itself. However, for future work, a CUDA-based *in situ* calculation of network measures and/or algorithms directly on the GPU seems to be a promising direction.

For future work, we plan to extend the flexibility and performance of the tool handling temporal and ensemble networks and to update the GPU-based implementation based on current and emerging technologies (e.g., compute shaders might simplify the handling of GPGPU-based methods and re-implementation based on Vulkan could increase the flexibility and performance of the system). In addition, we see potential extending the tool to the requirements of further types of networks and applications (e.g., trading networks).

ACKNOWLEDGMENTS

The authors would like to thank HPI for enabling the research and development of this tool. The research was supported by the Federal Ministry of Education and Research of Germany in the framework of InfectControl2020: https://www.infectcontrol.de/en/ (Project No.: 03ZZ0812D), in the framework of climXtreme: https://www.climxtreme.net/, in project PIK_Change (Grant No. 01LS2001A), and the DFG research training group (GK 2043) "NatRiskChange." A.A. and N.M. were also supported by the COPREPARE project funded by DAAD and UGC.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Stefan Buschmann: Conceptualization (equal); Formal analysis (equal); Methodology (lead); Software (lead); Validation (equal); Visualization (lead); Writing – original draft (lead). **Peter Hoffmann:** Data curation (equal); Formal analysis (equal); Investigation (equal); Validation (equal); Visualization (equal); Writing – original draft (equal). **Ankit Agarwal:** Data curation (equal); Formal analysis (equal); Investigation (equal); Validation (equal); Writing – original draft (equal). **Norbert Marwan:** Formal analysis (equal); Funding acquisition (equal); Investigation (equal); Methodology (supporting); Supervision (equal); Validation (equal); Visualization (supporting); Writing – original draft (equal). **Thomas Nocke:** Conceptualization (lead); Formal analysis (lead); Methodology (supporting); Project administration (supporting); Software (supporting); Supervision (equal); Validation (equal); Visualization (equal); Writing – original draft (lead).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] A. Agarwal, L. Caesar, N. Marwan, R. Maheswaran, B. Merz, and J. Kurths, "Network-based identification and characterization of teleconnections on different scales," Sci. Rep. **9**, 8808 (2019).
[2] J. F. Donges, Y. Zou, N. Marwan, and J. Kurths, "The backbone of the climate network," Europhys. Lett. **87**, 48007 (2009).
[3] N. Boers, B. Goswami, A. Rheinwalt, B. Bookhagen, B. Hoskins, and J. Kurths, "Complex networks reveal global pattern of extreme-rainfall teleconnections," Nature **566**, 373–377 (2019).
[4] C. Tominski, J. F. Donges, and T. Nocke, "Information visualization in climate research," in *2011 15th International Conference on Information Visualisation* (IEEE, 2011), pp. 298–305.
[5] C. Stoll, S. Gumhold, and H.-P. Seidel, "Visualization with stylized line primitives," in *Proceedings of IEEE Visualization 2005 (Vis'05)* (IEEE, 2005), pp. 695–702.
[6] V. Petrovic, J. Fallon, and F. Kuester, "Visualizing whole-brain DTI tractography with GPU-based tuboids and LoD management," IEEE Trans. Vis. Comput. Graph. **13**, 1488–1495 (2007).
[7] A. Wolff, "Graph drawing and cartography," in *Handbook of Graph Drawing and Visualization*, edited by R. Tamassia (CRC Press, 2013), pp. 697–736.
[8] "Methods for multilevel analysis and visualisation of geographical networks," in *Methodos Series*, edited by C. Rozenblat and G. Melançon (Springer, 2013), Vol. 11.
[9] P. Rodgers, "Graph drawing techniques for geographic visualization," in *Exploring Geovisualization*, edited by J. Dykes, A. M. MacEachren, and M.-J. Kraak (Elsevier, 2005), pp. 143–158.
[10] T. Nocke, S. Buschmann, J. F. Donges, N. Marwan, H.-J. Schulz, and C. Tominski, "Review: Visual analytics of climate networks," Nonlinear Process. Geophys. **22**, 545–570 (2015).
[11] S. Schöttler, Y. Yang, H. Pfister, and B. Bach, "Visualizing and interacting with geospatial networks: A survey and design space," CoRR abs/2101.06322 (2021).
[12] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. van Wijk, J.-D. Fekete, and D. Fellner, "Visual analysis of large graphs: State-of-the-art and future research challenges," Comput. Graph. Forum **30**, 1719–1749 (2011).
[13] Y. Hu and L. Shi, "Visualizing large graphs," Wiley Interdiscip. Rev.: Comput. Stat. **7**, 115–136 (2015).
[14] G. G. Brinkmann, K. F. Rietveld, and F. W. Takes, "Exploiting GPUs for fast force-directed visualization of large-scale networks," in *2017 46th International Conference on Parallel Processing (ICPP)* (IEEE, 2017), pp. 382–391.
[15] G. Brinkmann, K. Rietveld, F. Verbeek, and F. Takes, "Real-time interactive visualization of large networks on a tiled display system," Displays **73**, 102164 (2022).
[16] H. Linsenmaier, see https://medium.com/rapids-ai/large-graph-visualization-with-rapids-cugraph-590d07edce33 for "Large Graph Visualization with Rapids Cugraph" (2020) (accessed 3 December 2020).
[17] K. Zhou, Q. Hou, R. Wang, and B. Guo, "Real-time KD-tree construction on graphics hardware," ACM Trans. Graph. **27**, 1–11 (2008).
[18] B. Alper, S. Sümengen, and S. Balcisoy, "Dynamic visualization of geographic networks using surface deformations with constraints," in *Proceedings of the Computer Graphics International Conference (CGI)* (IEEE Computer Society, 2007).
[19] Y. Yang, T. Dwyer, B. Jenny, K. Marriott, M. Cordeil, and H. Chen, "Origin-destination flow maps in immersive environments," IEEE Trans. Vis. Comput. Graph. **25**, 693–703 (2018).
[20] B. Bach, E. Pietriga, and J.-D. Fekete, "Graphdiaries: Animated transitions and temporal navigation for dynamic networks," Trans. Vis. Comput. Graph. (TVCG) **20**, 740–754 (2013).
[21] C. Hurter, O. Ersoy, S. Fabrikant, T. Klein, and A. Telea, "Bundled visualization of dynamicgraph and trail data," IEEE Trans. Vis. Comput. Graph. **20**, 1141–1157 (2014).
[22] M. Botsch, M. Spernat, and L. Kobbelt, "Phong splatting," in *Proceedings of the First Eurographics Conference on Point-Based Graphics, SPBG'04* (Eurographics Association, Goslar, 2004), pp. 25–32.
[23] N. Boers, B. Bookhagen, H. M. J. Barbosa, N. Marwan, J. Kurths, and J. A. Marengo, "Prediction of extreme floods in the eastern Central Andes based on a complex networks approach," Nat. Commun. **5**, 5199 (2014).
[24] L. M. Casanova, S. Jeon, W. A. Rutala, D. J. Weber, and M. D. Sobsey, "Effects of air temperature and relative humidity on coronavirus survival on surfaces," Appl. Environ. Microbiol. **76**, 2712–2717 (2010).
[25] N. Watts, M. Amann, N. Arnell, S. Ayeb-Karlsson, K. Belesova, M. Boykoff, P. Byass, W. Cai, D. Campbell-Lendrum, S. Capstick, and J. Chambers, "The 2019 report of the Lancet Countdown on health and climate change: Ensuring that the health of a child born today is not defined by a changing climate," The Lancet **394**, 1836–1878 (2019).
[26] A. C. Lowen, S. Mubareka, J. Steel, and P. Palese, "Influenza virus transmission is dependent on relative humidity and temperature," PLoS Pathog. **3**, e151 (2007).
[27] F. Brenner, N. Marwan, and P. Hoffmann, "Climate impact on spreading of airborne infectious diseases," Eur. Phys. J. Spec. Top. **226**, 1845–1856 (2017).
[28] F. Brenner and N. Marwan, "Change of influenza pandemics because of climate change: Complex network simulations," Rev. d'Epidemiol. Sante Publique **66**, S424 (2018).