REALTIME MULTISCALE VISUALIZATION OF FLIGHT DATA

TIJMEN KLEIN

2013

[Revision 7fcca48, Sat Nov 23 13:52:52 2013 +0100]

Tijmen Klein: *Realtime Multiscale Visualization of Flight Data,* © 2013

SUPERVISORS:
Prof. dr. Alexandru C. Telea
Dr. Michael H.F. Wilkinson

[Revision 7fcca48, Sat Nov 23 13:52:52 2013 +0100]

ABSTRACT

In this thesis we present a set of multi-scale visualization techniques, with implementation, for air traffic data. The goal of our visualizations is showing attributes of spatio-temporal data for multiple levels of detail both in geographical and temporal space, which can help with finding anomalies in air traffic and optimizing flight paths. We allow continuous transitions between the visualization of instantaneous flight positions and visually aggregated data for multiple days. We use animation, bundling and density maps to reduce clutter and achieve visual scalability. Implementing our algorithms on the GPU allows us to visualize thousands of trails with millions of data points at interactive framerates.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

With the availability of relative cheap GPS-receivers and Universal Access Transceivers, the amount of generated movement data has vastly increased over the recent years. Movement data can be generated from vehicles (e. g. cars, ships, aircrafts), mobile phones, or even animals [Adrienko and Adrienko, 2011].

Within this field of movement data, we will focus on flight data, which is typically used for Air Traffic Control (ATC) and Air Traffic Management (ATM). ATC focuses on the organization of the current flow of air traffic to prevent accidents and minimize delays. ATM on the other hand analyses historical flight data to improve future air traffic flow. ATC and ATM use visualizations of air traffic for finding and explaining outliers, understanding correlations between congestions and weather patterns, training of air traffic controllers, and planning of flight routes to minimize delays and fuel consumption. Figure 1 shows the graphical display of a typical ATM system.

Movement data, as the name implies, consists of spatio-temporal measurements, and can optionally contain other attributes (e. g. velocity, status, callsign). Data points belonging to the same object can be grouped into a *trail* and one or more trails can form a movement dataset. Gathering, visualizing, and analyzing this data is a challenging and interesting field.

Visualizing movement data poses challenges for both visual and computational scalability. Since movement datasets are larger than their static variants, and can typically contain thousands of trajectories with millions of data points. Visualizing this in an interactive and realtime fashion is a computational challenge. At the same time, drawing thousands of paths on a computer screen will naturally result in visual clutter and occlusion, especially in high-density regions. Visualizing additional attributes further increases this problem.

ATC and ATM users analyze air traffic data for the following goals:

1. understand past incidents and improve safety;
2. asses new safety systems and the resulting aircraft routes;
3. reorganize the airspace and corresponding procedures to handle increasing air traffic;
4. understand weather effects on air traffic routes;
5. study profitability from a business viewpoint (number of aircrafts on a given route, busyness of airports);

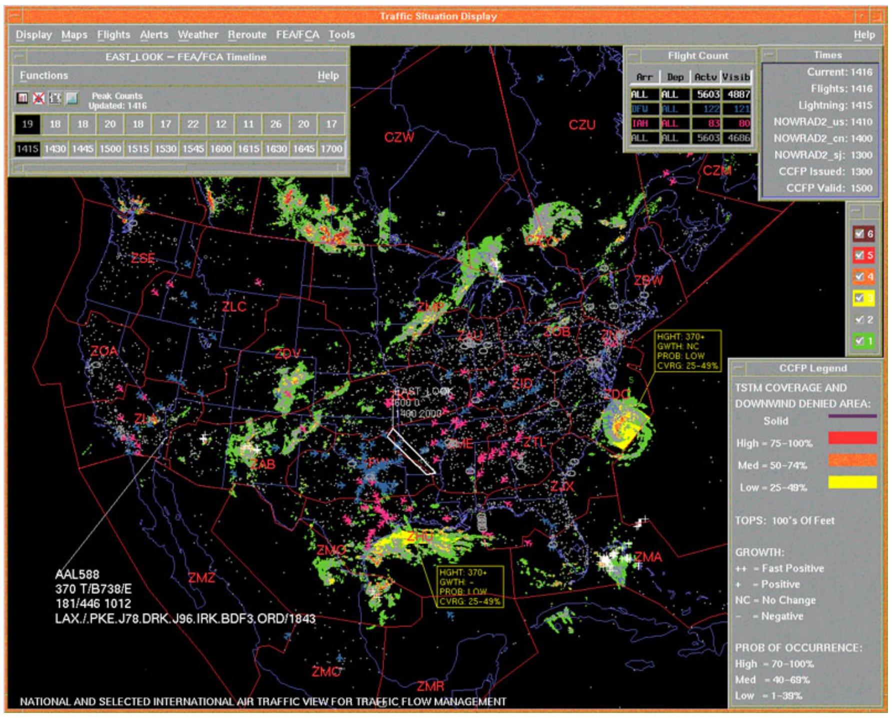


Figure 1: The Enhance Traffic Management System (ETMS) is an ATM tool used by the Federal Aviation Administration (FAA) to display air traffic, weather, alerts and statistical information.

6. examine trajectories for training purposes.

Furthermore, an application that assist with these tasks should be:

1. scalable in term of data size, number of supported attributes and computational speed;
2. easy and intuitive to use.

In this thesis we present a set of visualization techniques for multivariate attributed time-dependent air traffic data that addresses the above mentioned ATC and ATM goals. In contrast to existing ATC and ATM systems that focus on instantaneous flight positions or aggregated (textual) data, we focus on the visualization of aircraft trails with attribute data at different spatio-temporal levels of detail. Our techniques allow continuous navigation between instantaneous plane positions for a given time and visually aggregated trails over a time window of multiple days.

The structure of this thesis is as follows. Chapter 2 discusses related work in the field of movement data analysis (with a focus on air traffic visualization) and related visualization and filtering techniques. Our visualization techniques are explained in Chapter 3, while Chapter 4 focuses on their implementations. Chapter 5 shows some visual results for our techniques on different dataset and discusses the performance and limitations. Finally, we conclude the thesis in Chapter 6 and propose some future work.

PUBLICATIONS

Some ideas and figures from this thesis have appeared previously in the following publications:

- Christophe Hurter, Ozan Ersoy, S Fabrikant, Tijmen Klein, and Alexandru Telea. Bundled visualization of dynamic graph and trail data. *IEEE Transactions on Visualization and Computer Graphics*, 99(PrePrints):1, 2013b. ISSN 1077-2626. doi: http://doi.ieeecomputersociety.org/10.1109/TVCG.2013.246.
- Tijmen Klein, Alexandru Telea, and Matthew van der Zwan. Dynamic multiscale visualization of flight data. In *VISAPP (1), 2013*.

2 RELATED WORK

This chapter gives an overview of work relevant for our research. In order to gain insight in flight traffic visualization, we analyze existing Air Traffic Control systems and some state-of-the-art visualizations of different movement data (e. g. ships, cars) in Section 2.1.

2.1 MOVEMENT DATA ANALYSIS

This research focuses on the visualization of aircraft movement data. In this context, movement data consists of several trajectories of objects (e. g. aircrafts, ships), where each trajectory, or trail, is a temporal sequence of spatial data, optionally enriched with other attributes (e. g. altitude, velocity, direction, origin/destination). These spatio-temporal datasets can easily contain thousands of trails and millions of data points. An option to gain insight in this data is to use visualizations, either on the raw data or on aggregated forms of the data.

2.1.1 *Air Traf c Systems*

With a continuously increasing amount of air traffic, the need to regulate and analyze this traffic is larger than ever. The deregulation of air traffic makes this task even more complex. This is the reason for the development of Air Traffic Control (ATC) Systems: tools that help air traffic controllers with real time decision making or analysis and exploration of historical flight data.

There are some well known commercial-grade ATC Systems currently in use in the market. The Future ATM Concept Evaluation Tool (FACET) [Bilimoria et al., 2001] is an air traffic simulation and analysis tool developed by the NASA Ames Research Center. The purpose of FACET is to provide a simulation environment for the exploration, development and evaluation of ATM techniques. It uses aircraft performance profiles (e. g. cruise speeds climb and climb/descend rates), weather data, flight information and airspace models to present a 4D (spatio-temporal) flight simulation to its users. Different 2D and 3D views (see Figure 2) are used to display this air traffic data under simulated or measured conditions.

In addition to being a simulation tool, FACET is currently also used as an operational tool with real-time data. Federal Aviation Administration (FAA) traffic flow managers and commercial airline dispatchers have used FACET with integrated real-time data from the FAA radar systems and National Weather Service [Center, 2010]. Online

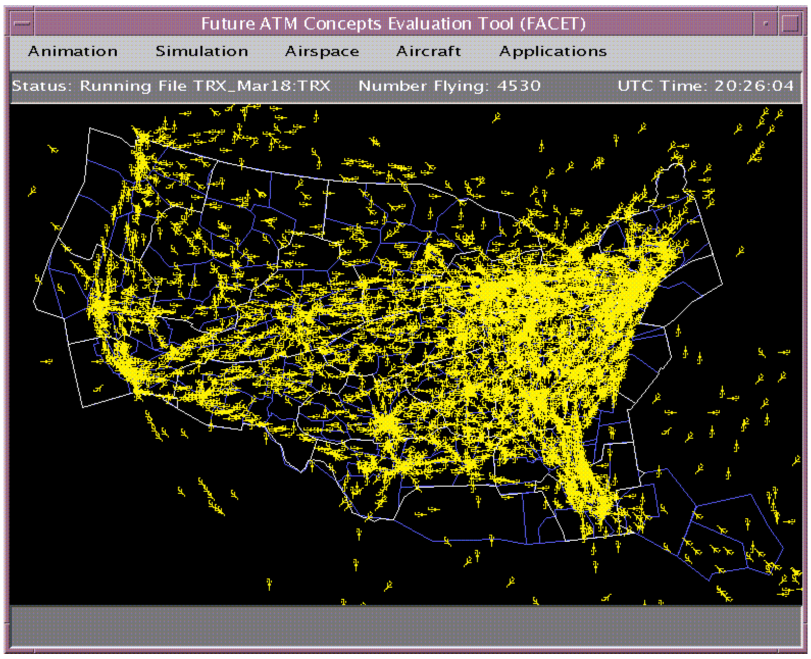


Figure 2: The Graphical User Interface of FACET showing a 2D view of air traffic data. Aircrafts are shown at their instantaneous positions using glyphs

videos [1] show some examples of FACET visualizing USA air traffic at various times (including September 11, 2001).

NEST [Eurocontrol, 2013] is a similar tool that has been developed by EUROCONTROL, the European Organisation for the Safety of Air Navigation. It is used internally by EUROCONTROL and Air Navigation Service Providers and focuses on airspace structure design and development, strategic traffic flow organization, scenario preparations and real-time simulations. It can use historical trajectory information or calculated trajectories based on known route networks for aircrafts. NEST is used to optimize the available resources (e. g. runways, airspace, aircrafts).

Epoques [Gaspard-Boulinc et al., 2003] (see Figure 5) is developed based on the requirements from the five French Air Traffic *en route* centers, and focuses on methods and tools for ATM Safety Occurrences, based on instantaneous flight position analysis. It gathers radar and audio recordings to help operators with detecting and analyzing air traffic incidents (e. g. aircrafts who went beyond their safety distance). These incidents happen frequently (in contrast to accidents), and serve as a useful metric to optimize flight routes.

Thales, Inc. is currently working on CoFlight, a next-generation flight data processing open-architecture framework for the storage, analysis, and visualization of spatio-temporal flight data [Thales, Inc., 2013].

1 http://www.youtube.com/watch?v=8pYiC7bTUxQ

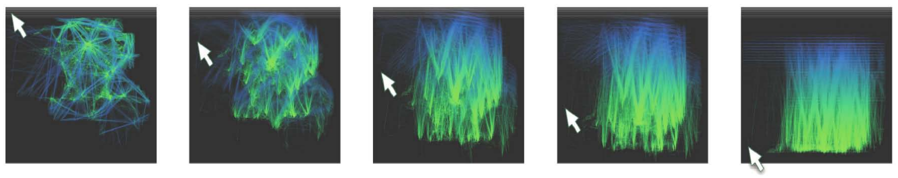


Figure 3: Users can control the transition from a top-down view (latitude, longitude) to a side-view (altitude, longitude) in FromDaDy by dragging the mouse

One of the interesting new tools that has been developed is FromDaDy, a visualization tool that focuses on the challenge of representing and interacting with aircraft trajectories that involve uncertainties [Hurter et al., 2009]. FromDaDy uses scatterplots, brushing, pick and drop, juxtaposed views and rapid visual configuration to explore multidimensional data. It allows the user to spread the data over several *linked views*. These views are based om complex interactive queries involving addition, removal and filtering of trajectories. Figure 4 shows the "pick and drop" interaction that uses a brushing technique to select data, which can be picked up and isolated for further inspection in a different view. The combination of these views can help the user to gain new insights in the flight data. The usage of multiple views makes this tool more visually scalable than previously mentioned tools. Figure 3 shows the interactive transition between one view (top-down) to a different view (side-view) in FromDaDy.

Many other ATC systems have been developed, a comprehensive overview of such systems is given by GAIN Group [GAIN Group, 2004].

While the previous mentioned systems and currently developed systems emphasize the importance of visualization for ATC systems, they all have very simplistic visualizations and focus on either instantaneous positions of aircrafts or the aggregation of all aircrafts. Specifically, there is no way to continuously navigate between these different levels of abstractions, which makes it harder to link coarse and fine patterns.

2.1.2 *Vessel Visualization*

Marine traffic shares similarities with aircraft traffic: both have great freedom of movement (compared to road constrained vehicles), are equipped with GPS broadcasting equipment, and are constrained to safety regulations. Therefore it makes sense to find inspiration in existing vessel visualization techniques. Willems et al. have developed a vessel visualization that enables operators of coastal surveillance systems to easily find the significant maritime areas and routes (anchor points and "highways") [Willems et al., 2009]. Their visualization is based on Automatic Identification System (AIS) data: a systems that is

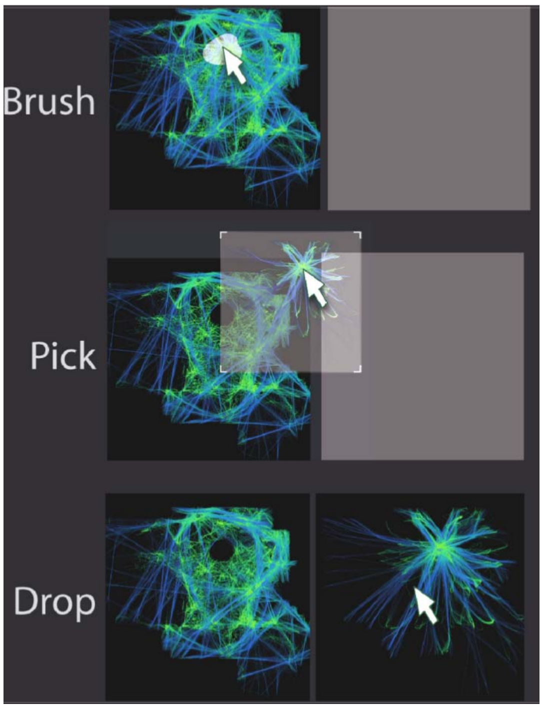


Figure 4: The "pick and drop" interaction in FromDaDy allows the user to use a brush to select data and isolate this selected data in different view

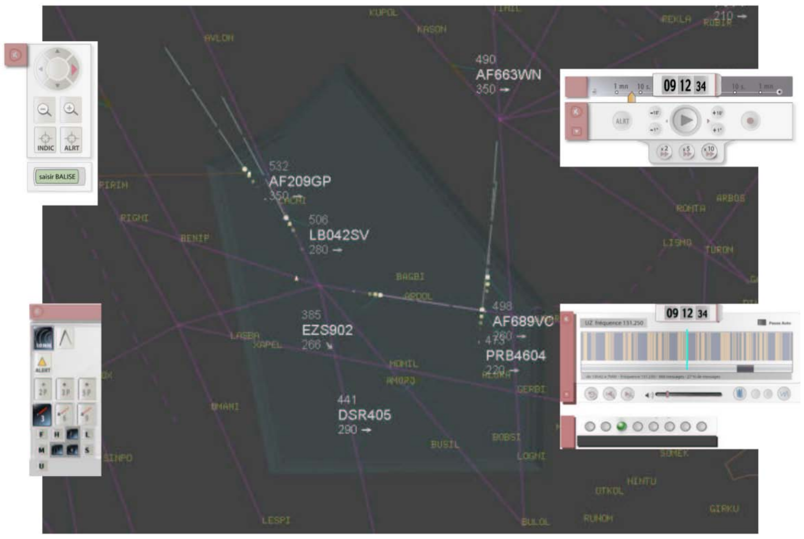


Figure 5: The Epoques system focuses on the instantaneous position of planes based on radar and audio recordings

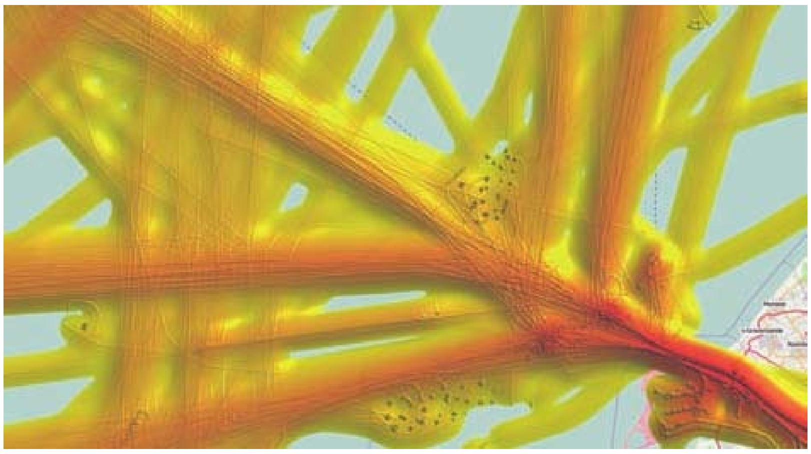


Figure 6: Multi-scale Vessel density on a logarithmic scale in front of the harbour of Rotterdam, The Netherlands, of a single day convolved with kernels of 100m and 1.5km.

currently equipped on my vessels; it exchanges data (unique identification, navigation status, position, speed, direction) with other nearby ships, AIS base stations and satellites [ITU, 2001]. Thousands of these nautical trajectories are visualized using density fields that are computed from a convolution of the dynamic vessel positions and a kernel. The density field is shown as a illuminated height map (see Figure 6). A combination of a large and small kernel is used to provide overview and detail; a large kernel provides overview and is able to show vessel highways, while a small kernel can focus on the instantaneous positions of vessels. Willems et al. compared their method against well-known trajectory visualizations (e. g. animation of moving dots) and tested these visualizations with tasks that are common for maritime analysis: find stopping vessels, fast moving vessels and estimate the busiest routes. They claim that their methods is the best for finding stopping objects and performs equally to the other methods for the other tasks [Willems et al., 2011]. The overview and detail combination that is provided by this method would be a nice addition to existing ATC systems. However, it is limited to static trajectory analysis (no animation) and not capable of quickly changing the level of detail or overview.

Riveiro and Falkman argue that visualization and user input are the critical challenges for maritime anomaly detection, since fully autonomous anomaly detection is rarely used in the real world due to their complexity and high amount of false alarms [Riveiro and Falkman, 2011]. Anomaly detection should therefore focus on providing adequate support for the human decision makers. This can be confirmed by the fact that multiple coastal administrations have collaborated with scientist to develop visual analytics systems that aid with the interpretation of historical AIS data. The Norwegian Coastal Administration had specific questions about potential shipping lane optimizations and used a Kernel Density Estimation (KDE) based technique to help them answering these questions [Lampe et al., 2010]. The U.S. Coast Guard's Ninth District and Atlantic Area Commands worked together with Malik et al. to develop a system that helps with the analysis of historic response operations and assessment of the potential risks in the maritime environment. This shows that air traffic visualizations should probably follow a similar route.

2.1.3 *Other Movement data*

Aircrafts and vessels make up for most of the current movement visualizations, which can be explained by the ease of traceability due to required GPS equipment and economical impact. However, all previously mentioned techniques can also be applied to other forms of movement data (e. g. cars, scooters, animal migrations, tourist routes). Adrienko and Adrienko show that reducing visual clutter by means of abstraction and spatial generalization can be an effective tool in massive movement visualization [Adrienko and Adrienko, 2011].
Their method divides the trajectories into aggregate flows between areas. These areas are not predefined, but are calculated using significant points from the trajectories. While an effective way to show a course flow of routes, this method discards other attributes and oversimplifies the trajectories.

Searching for the diversity of routes with a fixed origin and destination (based on taxi routes) again shows the importance of clutter reduction and information filtering [Liu et al., 2011]. The analysis of route diversity can be used to find potential bottleneck for transportation management.

MoleView is a technique for interactive exploration of multivariate data with a spatial embedding. It defines a semantic lens which selects a specific spatial and attribute-related data range, and keeps the selected data in focus while continuously deformed the unselected data to maintain the context [Hurter et al., 2011]. TrajectoryLenses is an interaction technique that extends the exploration lens metaphor to support complex filter expressions and the analysis of long time periods. It allows both temporal and spatial filters, and can be used with interactive map to find geospatial areas of interest. Combining different lenses allows for more complex and aggregated queries [Krüger et al., 2013]. These focus + context techniques can be an important tool in reducing the visual clutter even further.

2.2 CONCLUSION

The systems mentioned in this chapter do not fulfill all requirements that ATC users have. First, the ability to handle and visualize multivariate data is often limited. Systems focusing on the instantaneous positions of aircrafts have very limited (visual) scalability. And finally, many of the analyzed systems suffer from large amounts of visual clutter. Therefore, we have developed a set of multi-scale visualization techniques for air traffic data. These techniques help to reduce clutter and achieve visual scalability.

3 IMAGE-BASED VISUALIZATION

This chapter introduces our image-based visualization techniques for aircraft trails, based on the requirements mentioned in Chapter 1.

3.1 OVERVIEW

This section provides a bird's-eye view of our visualization techniques and pipeline, which is illustrated in Figure 7

We have used 2 data sources for the development and testing of our application. One originates from the French ATC authorities and contains one week of flights above the French airspace. Our second dataset is scraped from the Planefinder [PlaneFinder, 2013] website, and contains a month of worldwide user-collected flight data. Both of these datasets consist of a collection of *aircraft trails*, where each trail contains an ordered sequence of points along the flight path. Our main visualizations are based on the concept of a *sliding time-window*, which selects all points within a given time-range. The visualization components within our system use this sliding time-window to construct an animated visualization. The last component adds interaction to our system, which allows the user to directly manipulate the visualization with instant feedback.

Section 3.2 further explains the datasets and how we can formally model the data and how this data relates to dynamic graphs. Our aggregated trail visualization, based on density maps, is introduced in Section 3.3. We further extend this visualization on the temporal-scale in Section 3.4. Section 3.5 discusses our enhanced color mapping techniques based on transfer functions. In Section 3.6 we introduce various bundling techniques for static and dynamic graphs, and how we can apply bundling to our datasets. Our visualization to find congested flight areas is explained in Section 3.7. Finally, we conclude this chapter in Section 3.8.

3.2 DATA

The flight visualization application discussed in this thesis is developed around 2 datasets.

3.2.1 *Radar recordings*

The first is the data used by Hurter et al. and consists of radar recordings above the territorial airspace of France, and is obtained directly

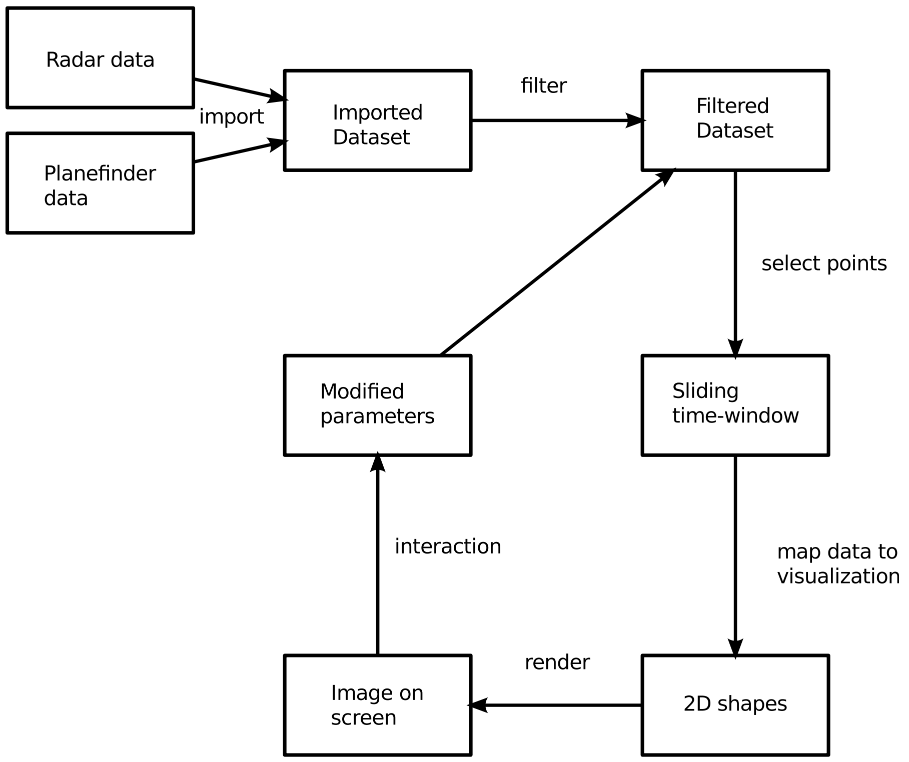


Figure 7: System diagram of our system

from the French ATC authorities. This data is recorded between the 6th and 12th of april 2008 (one week) and contains 52,547 flights with a total of 870,880 sample points. Since it is based on radar information, each sample point only contains the basic minimal information: flight id, position (latitude and longitude) and altitude. Other attributes (e. g. speed and direction) can be calculated from this information.

3.2.2 *Plane nder*

The second dataset is originating from `http://planefinder.net/`, a website that continuously gathers Automatic Dependent Surveillance Broadcast (ADS-B) plane feeds [PlaneFinder, 2013]. ADS-B is used by aircrafts to transmit their identification, position, altitude, velocity, callsign and status to other airplanes, satellites and ADS-B base stations [ADS-B, 2013]. The workings of ADS-B are explained in Figure 8: aircrafts communicate with GPS satellites to determine their own position, and use on-board equipment to measure their altitude and velocity. This information, bundled with their identification and callsign, is broadcast with the ADS-B *out* equipment. This information is not only picked up by base stations on the ground, but also by communication satellites and other aircrafts, which helps to increase airspace security.

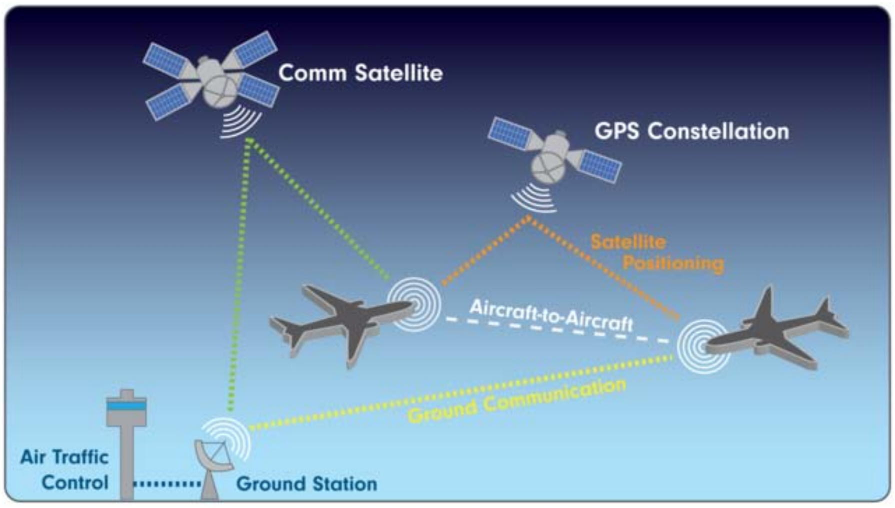


Figure 8: Aircrafts use GPS satellites to determine their position. ADS-B equipment is used to broadcast this position to other airplanes, communication satellites and ADS-B base stations.

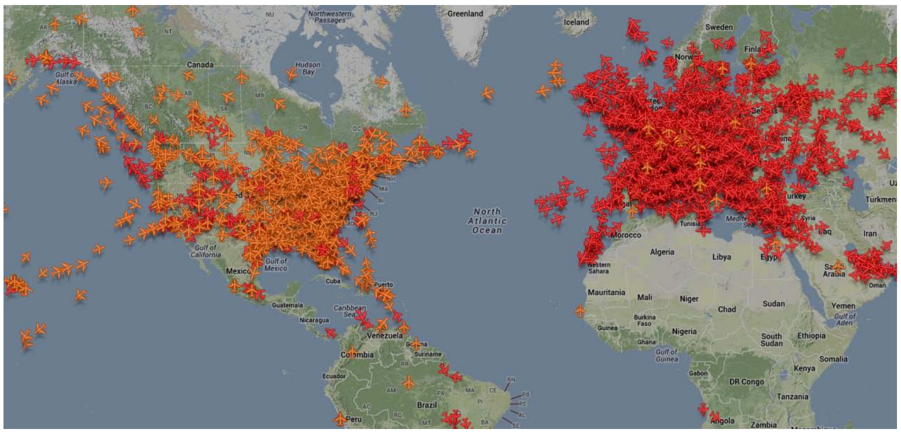


Figure 9: Planefinder showing worldwide aircraft positions at a given time.

Base stations are often placed by governments and ATC's, but anyone is free to listen to ADS-B signals. Hobbyist with ADS-B equipment can submit their gathered information to projects like Planefinder, which acts as a central server and merges all gathered data. The Planefinder website displays this gathered information, but is only capable of showing the instantaneous plane positions at a given time (see Figure 9). This form of display is only useful with a low density of flights.

ADS-B is very similar to the AIS systems used by ships discussed in Section 2.1.2, and is gradually replacing radar as the most efficient method for air traffic control. ADS-B is more precise than radar both in temporal as in spatial aspect since it broadcasts every second and uses GPS for positioning.

Most commercial aircrafts are already equipped with ADS-B equipment, and this will only increase in the future. The United States is rolling out a network of ADS-B base stations and will require all aircrafts flying in airspaces of class A, B or C to broadcast via ADS-B by 2020 [Department of Transportation, 2010]. The European Union is working on similar rules: aircrafts with a weight above 5700 kilograms or a maximum cruise speed above 250 knots that want to fly in the *single European sky* will be required to have ADS-B equipment from 2017 [EU, 2011]

We gathered one month of data (June 2013) from the Planefinder website, which contains 748,057 flights with a total of 14,711,646 sample points.

3.2.3 *Data model*

In order to compare our data with similar datasets, relate to other work, and discus possible visualization methods, we model the previously introduced datasets at a more fundamental level. The path of a single flight, or trail, can be modeled as sequence of points $\mathbf{p}_i$

$$T = \langle \mathbf{p}_i = ((\textit{latitude}, \textit{longitude}) \in \mathbb{R}^2, t \in \mathbb{R}^+, a^n \in \mathbb{R})_i \rangle \qquad (1)$$

which is ordered along increasing values of t_i, which depicts time. Next to the latitude, longitude and time, each point $\mathbf{p}_i$ holds an attribute vector a^n, where n is the number of additional attributes per point. Examples of additional attributes are height, speed and direction. Attributes can either be measured (e. g. flight id, height) or derived (e. g. acceleration, climb rate). These attributes can vary per sample point (e. g. height, speed), or per trail (flight id, airplane model, average speed).

At a global datamodel we have a multivariate trail-set

$$TS = \{T_j\} \qquad (2)$$

that consists of a collection of trails. This trail-set can be considered as a particular case of a multivariate dynamic graph [Hurter et al., 2013]. If we consider the start and end point of a trail as nodes, and the trail itself as an edge, then our dataset can be considered as a graph. Naturally, edges have a lifetime $[t_{min}, t_{max}]$ (the period in which this flight is active) and a duration $d = t_{max} - t_{min}$ (the duration of the flight). Nodes also have a lifetime, based on the lifetimes of its corresponding edges. Note however, if we would convert TS to a graph, none of the nodes would have more than a single edge connected to it. This is because the data in TS does not have any explicit information about shared end-points (airports). A distance-based clustering could in theory group nearby end-points (nodes) into a single node. However, this would be a change of data and not a fully automatic operation. The outcome of this operation and the original TS would not be interchangeable.

The attribute vector a^n of each point maps to the edge attributes of a graph. However, edge attribute do not change along the edge in a typical graph, while our point attributes can vary per sample point (e. g. height, speed). This makes edge attributes not very well suited for our per-point varying attributes. Our trail-based attributes (flight id, average speed) on the other hand, map naturally to edge attributes.

It is also possible to compute node-attributes based on TS, end-notes get the aggregated values of their matching trail-attributes. This could, for example, be the average duration of flights. However, in order to make note-attributes useful, we would have to apply the previously mentioned clustering to group geographically close nodes. Otherwise, node-attributes would be identical to the attributes of their single edge.

In terms of time, temporal graphs are known for having two variations: sequence graphs and streaming graphs [Hurter et al., 2013]. A sequence graph consists of a time-ordered set of graphs $G^i = (V^i, E^i)$, where each graph G^i contains a snapshot of a time-dependent system at time i, for a total of N moments in time. Streaming graphs, on the other hand, are defined on a vertex set V and edge set E, where edges are defined by their lifetime and nodes. Streaming graphs are a natural form when there is no predefined sequence or when dealing with live data sources [Andrienko and Andrienko, 2005]. Unfortunately, sequence and streaming graphs do not allow interchangeable visualizations. Our trail-set TS is much more like a streaming graph than a sequence graph, since we have no notion of discrete snapshots.

3.3 AGGREGATED TRAIL VISUALIZATION

Drawing all trails in the trail-set TS is a viable solution for only the most trivial (and sparse) versions of TS. In more realistic cases, drawing everything from TS at once will result in a lot of visual clutter, and limit our visual scalability. Therefore, we propose a multiscale visualization. Our first scale parameter is the spatial one. We use a scale-space approach that groups trails which are spatially close and numerous into a single visual abstraction. Similar to Scheepens et al. and Willems et al. we use density maps as the prime ingredient for this visualization. However, in contrast to previous work, in order to achieve a dynamic temporal visualization we use animation. All our visualization techniques are centered around a time $t_{min} \leqslant t_{current} \leqslant t_{max}$ that always falls within the range of times defined in TS, and every animation step $t_{current}$ is increased with a user defined animation speed t_{speed}.

One way to aggregate this data is using density estimation, which can be displayed as a density map [Andrienko et al., 2011, 2012; Marzuoli et al., 2012]. A popular density estimator is the Kernel

Density Estimation (KDE) [Silverman, 1986; Rosenblatt, 1956; Parzen, 1962], which given a series of samples x from a distribution with density f can be estimated as:

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{3}$$

where K is the kernel and h is the smoothing parameter (or bandwidth). Typical kernel choices are Gaussian functions and Epanechnikov (quadratic) functions.

Lampe and Hauser use KDE to display an interactive density map with streaming data from various sources (AIS ship data, monetary contributions to the Obama campaign, traffic in the Bay Area, air traffic above the USA). Due to their GPU-based implementation of KDE, they are able to achieve interactive framerates. Implementing our density map on the GPU is therefore an obvious choice to achieve computational scalability.

In order to provide contextual information, we first define the collection

$$TC_1 = \{T \in TS \mid T_0(t) \leqslant t_{current} \leqslant T_n(t)\} \tag{4}$$

as all trails T that contain $t_{current}$ and create a density map

$$\rho(\mathbf{x}) = \sum_{T \in TC_1} \int_{\mathbf{p} \in T} K\left(\frac{\mathbf{x} - \mathbf{p}}{h}\right) \tag{5}$$

by convolving the selected trail-collection with a kernel K of width h. Similar to the work of Scheepens et al. and Willems et al. we display the density ρ directly. However, in contrast to previous work, this density map only serves as contextual information in our application. Fine-grained visualization can later be added on top of this density visualization to provide focus. In order to serve as context and maintain distraction free, ρ is mapped to a black-to-white (luminance) colormap that is drawn below all other elements.

Figure 10 shows the density map for the French airspace dataset, and displays the full trails for all flights that were active at a certain time. Bright areas indicate places where a lot of this currently active air traffic passes. Because of the definition of the selected plains and the construction of the density map, a location where 2 trails cross each other will show up brighter than the rest of the trail, although the passing does not necessarily need to happen at the same time. Darker areas indicate regions where few flights are active during this time, while black areas means that there were no aircrafts at all. This Figure already clearly shows some dense flying areas near larger cities (e. g. Paris, Toulouse, Bordeaux, Lyon); but also, interestingly, dense flying *routes* such as between Toulouse and Lyon.

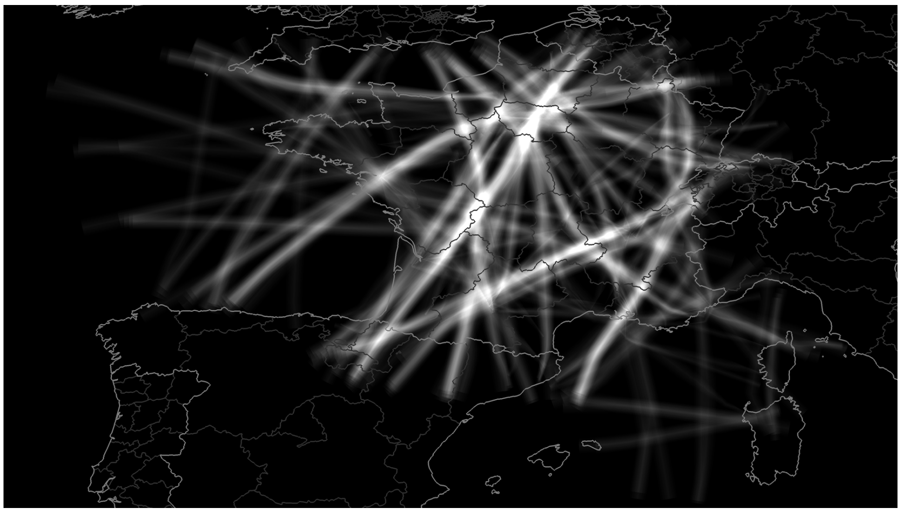
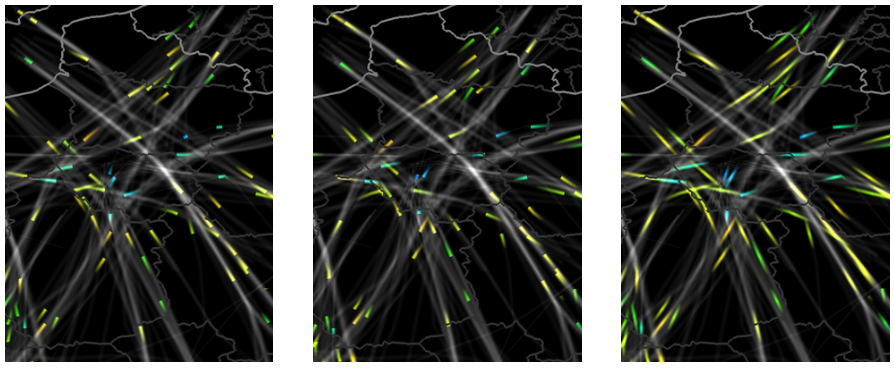


Figure 10: Contextual density map for the French airspace dataset using a black-to-white grayscale colormap

Clearly, spatial aggregation works very well for removing high-frequencies and similar-scale clutter, while maintaining the coarse-scale occupancy patterns. We can clearly distinguish the dense flying routes and airports, and thus answer questions like: "which spatial zones have been most flown over during the period of interest?". Manipulating the filter-size allows us to control the degree of simplification. On the other hand, spatial aggregation is quite aggressive and may oversimplify our data. We lose the information of our attributes, this visualization only shows spatial density for a given time period. It is not possible to distinguish individual flight ids, height, direction and speed from this visualization. It is also hard to gain time-dependent insights based on this visualization. Therefore, we keep this visualization as a general overview technique, and add more details on top of it for more fine-grained insights.

3.4 MULTISCALE TIME VISUALIZATION

The visualization discussed in Section 3.3 is good for aggregating spatial information, but lacks information in the temporal domain. Therefore, this section introduces the second scale parameter, time, for our multi-scale visualization. We apply the same concepts from the previous section (filtering and aggregation) in the time domain, to show time-dependent information.



(a) $w(t) = [t, t + \Delta]$ (b) $w(t) = [t - \Delta, t]$ (c) $w(t) = [t - \Delta, t + \Delta]$

Figure 11: Modifying the sliding time window $w(t)$ to either show front, back or both sides of the trail segment. Height is mapped to color, and $\Delta = 2$ minutes. The *contextual density map* is used as a background.

Sliding Time Window

In order to expand our visualization, we first define a *sliding time window* $w(t) = [t, t + \Delta]$ which moves at constant speed t_{speed}. Next, similar to the contextual density map, we define the collection

$$TC_2 = \{T \in TS \mid T_0(t) \leqslant t_{\text{current}} + \Delta, t_{\text{current}} \leqslant T_n(t)\} \tag{6}$$

as all trails T that contain a point $\mathbf{p}_i \in w(t)$. Rather than drawing the whole trail, as we did with the contextual density map, we now focus on trail segments $T_\Delta(t)$ that contain only the part of T that fall within $w(t)$. These trails are drawn with a transparency (alpha) texture that is fully opaque at t and fully transparent at $t + \Delta$. This results in arrow-like shapes, that, when combined with the animation effect of the sliding winding $w(t)$ and a low value of Δ are capable of showing the instantaneous positions of the aircrafts and the paths along their respective trails (see Figure 11a).

The default behavior of $w(t)$ is to show the front of the trail. However, $w(t)$ can also be modified to show the back of the trail, or both parts, as is demonstrated in Figure 11. Using a sliding time window $w(t) = [t - \Delta, t]$ results in *reversed* arrows (comet-like shapes),that show the current position (similar to $w(t) = [t, t + \Delta]$) and the previous positions of the aircraft (see Figure 11b). Combining both $w(t)$'s results in $w(t) = [t - \Delta, t + \Delta]$ (see Figure 11c), which shows the instantaneous airplane position fully opaque and gradually becomes more transparent both in the future and past direction.

Due to the construction of the sliding time window $w(t)$, the velocity of the aircraft is reflected in the *length* of the trail segment. The trail segment shows the part of trail that is covered within the time period $w(t) = [t, t + \Delta]$, a long trail segment indicates that a large distance was covered during this time, while shorter trail segments indicate a short distance. Since the average velocity

$$\bar{v} = \frac{\Delta x}{\Delta t}$$

is the displacement (Δx) during a time interval Δt and all trail segments share the same Δt, we can conclude that the length of the trail segments correspond to the average velocity in the time window $w(t)$.

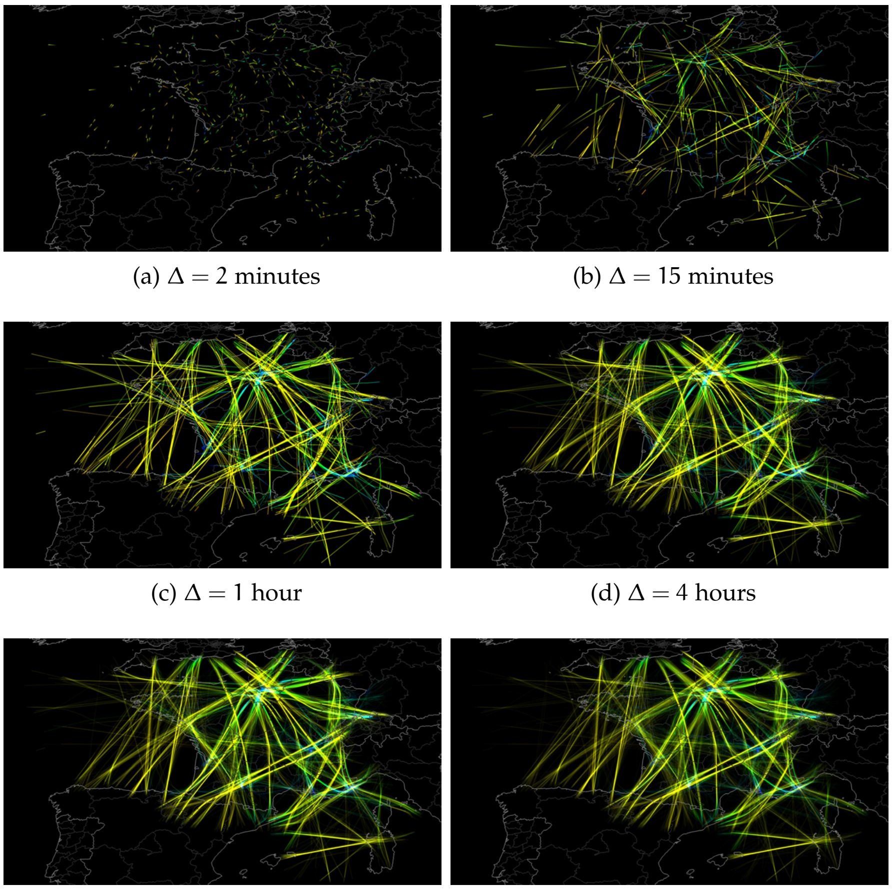
Smooth time-based ltering

By gradually increasing Δ we can smoothly transition from instantaneous aircraft position to more global and aggregated views. Additive alpha blending is used to enable visual aggregation of the different trail segments. This effect is demonstrated in Figure 12, which shows a continuously increasing Δ ranging between 2 minutes and 16 hours for our one week dataset above France. Due to the aggregation effect of longer trails, the contextual background trails are not necessary anymore; in fact, they would only increase clutter for large values of Δ. Low values of Δ enable us to pick out individual aircrafts and their respective attributes at the current time. Slightly larger values of Δ can be used to see *transitions* of attributes for flights, e. g. ascending or descending flights, which can be observed when carefully examining Figure 12b. Further increasing Δ means that we can no longer track individual flight paths.

However, it allows us to see global patterns in *routes* and *attributes*. Figure 12c to Figure 12f clearly show us that the aircrafts above France mostly fly in certain predefined paths during this time period. Using $\Delta = 1$ hour still shows individual outliers to these paths, while the higher aggregation factor of $\Delta = 16$ hours makes these outliers nearly invisible. Altitude is mapped to color in Figure 12, blue spots indicate low flights, while warm colors show us in-flight routes. Large blue zones in the aggregated images are near the large airports of France. Looking at the warm colored in-flight routes, we see little variation in cruising altitude, which correlates with the flight rules for civic aircrafts above France and optimal cruising conditions.

Transparency texture length

Next, we introduce a user-set parameter δ that determines the *length* of our transparency texture. Setting $\delta = \Delta$ would change nothing to our previous visualizations: the texture is stretched along the full length of trail segment that spans Δ. However, setting $\delta < \Delta$ results in a repeating texture along the trail segment, which visually looks like a “train” of short arrow-like trail segments. This effect is demonstrated in Figure 13, where $\delta = 1$ minute and $\Delta = 10$ minutes: each



(a) $\Delta = 2$ minutes

(b) $\Delta = 15$ minutes

(c) $\Delta = 1$ hour

(d) $\Delta = 4$ hours

(e) $\Delta = 8$ hours

(f) $\Delta = 16$ hours

Figure 12: Gradually increasing Δ allows smooth transitioning from instantaneous positions to aggregated trail segments. Altitude is mapped to color

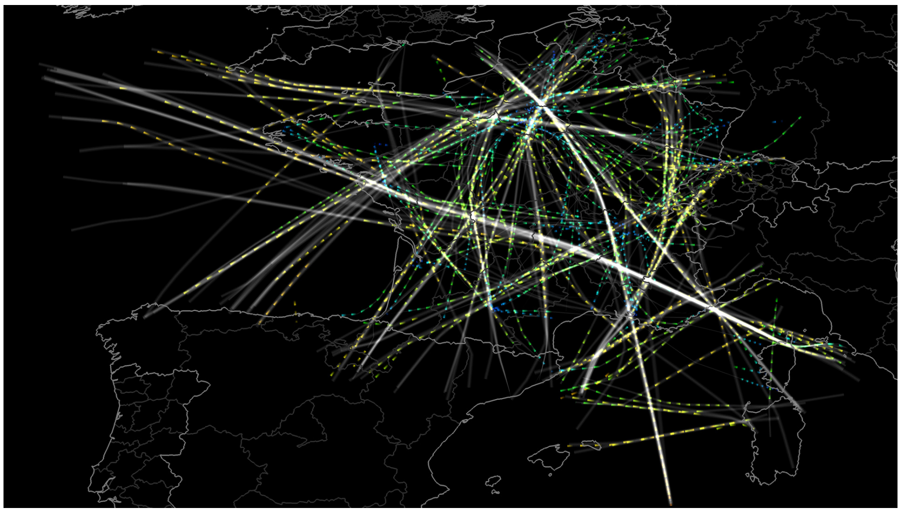


Figure 13: Short trail segments can be shown using $\delta < \Delta$.

transparency texture is repeated 10 times per trail segment. The advantage of this technique over $\delta = \Delta = 10$ minutes, is that the train of short trail segments is still capable of clearly showing a *direction*, due to the arrow-like shape.

These trails are color mapped with one of the attributes of the points $\mathbf{p}_i$, e. g. height or direction. Height is color mapped using a traditional rainbow color map (see Figure 11 and Figure 12), for direction however, this would be very hard to interpret. Therefore, we developed a directional hue based color map; hue naturally maps to an angle (as happens with a color wheel) and therefore is a logical candidate to map the directions of the airplanes. Figure 14 shows en example of such a color map, and allows us to easily find paths that are close to each other and parallel, but go in opposite directions (e. g. yellow/blue: northeast/southwest, pink/green: northwest-/southeast). From these images we can conclude that parallel but opposite flight routes often are close to each other, but do not overlap. We chose to use *additive blending*, white areas can therefore be interpreted as regions where aircrafts fly in many (or at least: opposite) directions. This happens, for example, above Paris; but interestingly, a white path can be seen between Mallorca/Minorca and Sardinia: apparently opposite flying planes share the same overlapping path for this route.

Conclusion

By applying two different filters (Δ and δ) in the temporal domain we are able to show dynamics at different time scales. We can show the emergence and disappearance of trails over specific time periods (e. g. hours or days), but also spatial occupancy of some region by trail fragments, up to the detail of the actual instantaneous positions

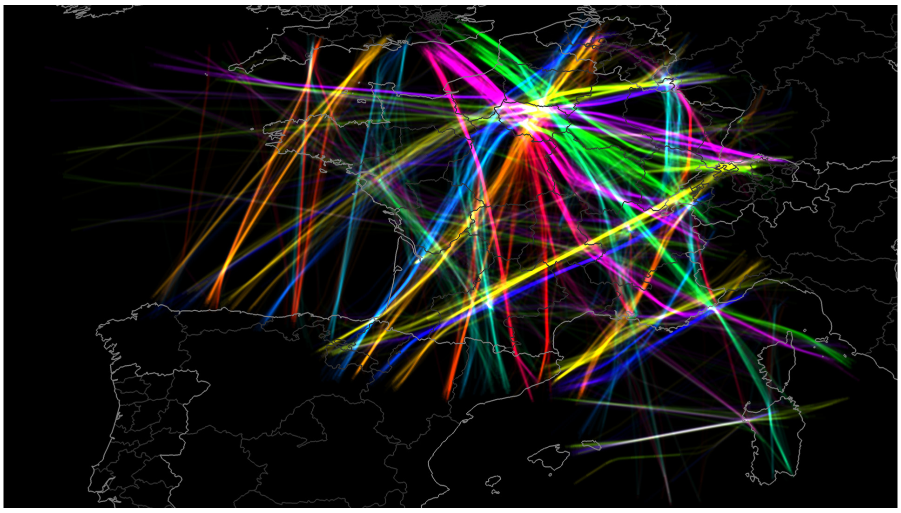


Figure 14: Directional color map above France using additive blending, $\delta = \Delta = 8$ hours

of airplanes. By using a combination of different visualization techniques, we are able to map 3 attributes at the same time. The velocity is encoded in the length and shape of the trail segments (1), the instantaneous aircraft positions are shown as their geological position above a map (2), and color can be used to map a remaining attribute (e. g. color) (3). Animation is used to show the changes of these attributes over time.

3.5 MAPPING ATTRIBUTES TO COLORS

In the previous parts of the chapter, we showed how color mapping is used to show different types of attributes. Section 3.3 uses grayscale mapping to show the spatial occupancy. Luminance is used, via color blending, in Section 3.4 to show the same occupancy signal, but at a different temporal scale. Moreover, texture (by luminance modulation) and hues are used to add supplementary data attributes, such as height and flight direction.

While our current color mappings are useful, they have to be further refined to become truly effective for the requirements of ATC users. There are a few reasons why these color mappings need optimizations. First of all, our datasets are large and have many occlusions. At these dense areas, color mapping has to be carefully designed to show the interesting aspects and not to convey false insights. Furthermore, the combination of animation and texturing creates more high-frequency signals than a colormapping based on luminance alone. Interpreting high-frequency signals can be challenging, so our color maps need to be optimized for these tasks.

Our challenge therefore is: how can we design a set of colormaps that are able to create clear, clutter-free visualizations, while showing the largest possible number of attributes at different time and space scales. This should happen in an intuitive way, so that events of interests are easy to spot.

Transfer Functions

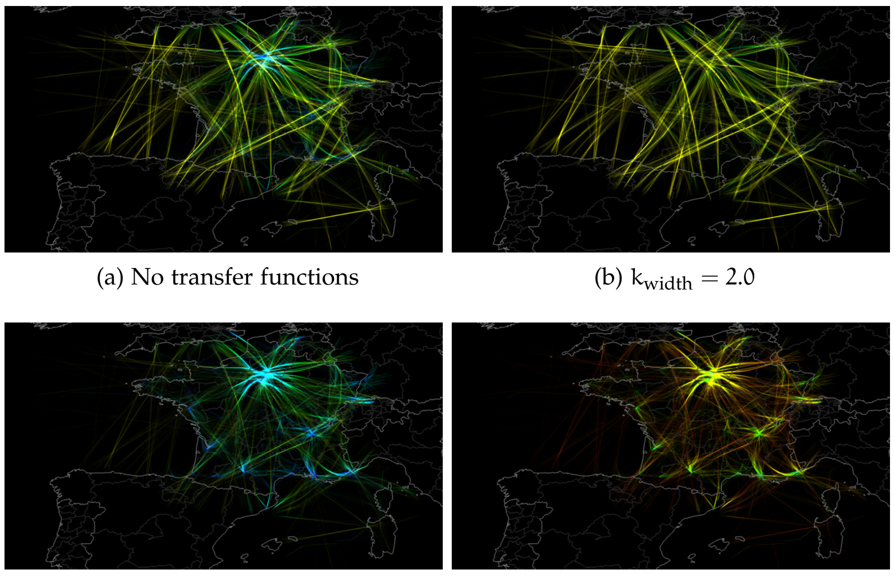
When using large values of Δ, we can aggregate information from multiple trails. However, this does come at a cost: we are no longer able to see individual aircrafts, and (therefore) not able to filter out specific trails. For example, when looking at the Paris area, we see all flights above this region and are not able to focus on either landing planes or fly-over planes. Focusing on these areas is crucial for certain ATC tasks [Letondal et al., 2013; Hurter et al., 2009]. While the first of these costs is a direct consequence of aggregation, we are able to offer a solution for the second problem by means of transfer functions.

First, instead of using a fixed kernel width, we introduce a parameter k_{width} that we use for our *width transfer function*: $f(w) = (\frac{h}{h_{max}})^{k_{\text{width}}}$, where h and h_{max} are the altitude and its maximum value respectively. Setting $k_{\text{width}} = 0$ renders all trail segments with the same width, while $k_{\text{width}} = 1$ linearly scales the width of the trail segment according to the height (aircrafts at ground level get a near 0 width, while aircrafts near the maximum height get maximum width). Setting $k_{\text{width}} < 0$ or $k_{\text{width}} > 0$ results in an exponential distribution between the minimum and maximum width.

Next, we introduce an *alpha transfer function*: $f_1(h) = (\frac{h}{h_{max}})^{k_\alpha}$ that is based on the user-set parameter k_α. Our pulse texture ϕ_i is modulated with $f(h)$. This allows for an alternative way (by means of transparency) to focus on higher altitude aircrafts. Alternatively, we can use $f_2(h) = (\frac{h_{max}-h}{h_{max}})^{k_\alpha}$ to focus on low altitude trail segments. Setting $k_\alpha = 0$ will not modify the original alpha level, while increasing $k_\alpha > 0$ will gradually render more trail-segments transparent, allowing us to focus on the values of interest.

Finally, our *color transfer function* allows us to apply more dynamic range to a certain range of the color mapped attribute. For example, when focusing on low altitude aircrafts with $f_2(h)$, we would also like to see the variation on height within this focused range. To achieve this, we apply the transfer function $f(x) = x^{k_{\text{color}}}$ to the normalized value of the attribute that we use for color mapping (in this example: altitude). Values of $k_{\text{color}} > 1$ emphasize high altitude ranges. Values $k_{\text{color}} < 1$, in contrast, emphasize low altitude ranges.

Figure 15 demonstrates how these transfer functions can be applied to our French airspace dataset. We use a fairly large $\Delta = 8$ hours in order to get some aggregation and a fair amount of clutter, and we map the height attribute to color. The basic result of this can be seen in Figure 15a, where we can see some low-altitude ranges (near Paris) and a lot of cruising routes. By setting $k_{\text{width}} = 2.0$ as in Figure 15b,



(a) No transfer functions (b) $k_{width} = 2.0$

(c) $k_{\alpha} = 2.0$, $k_{width} = 0.0$ (d) $k_{\alpha} = 2.0$, $k_{width} = 0.0$, $k_{color} = 0.25$

Figure 15: Applying transfer functions allows us to focus on specific altitude ranges

we decrease the width of the lower trail segments, which results in less cool-colored areas. This allows us, for example, to focus on the flights that fly above Paris, without being distracted by flights that land there. Figure 15c shows the effect of setting $k_{\alpha} = 2.0$. The high-altitude trail segments become more transparent, which makes it easier to focus on the airport zones (areas with low altitudes). These areas, containing many take-offs and landings, can be seen as cool colored (blue/green) areas. While the Paris area clearly shows many low-altitude traffic, we can now also spot other airports that were not apparent without transfer functions (e. g. near the coast). However, we can still not see much variance in height during the take-off and landing trajectories; almost all colors are in the "cool"-range. This can be solved by setting $k_{\alpha} = 2.0$ and $k_{color} = 0.25$, of which the results can be observed in Figure 15d. By applying more dynamic range to the low altitude flights, it becomes easier to see the ascending and descending of the aircrafts around Paris. Ascending and descending flights now use the blue-to-yellow part of our rainbow color map. Aircrafts at cruising altitude can still easily be distinguished due to their very warm (red) color.

By using a combination of different transfer functions, we are able to optimize our colormapping process for specific ATC tasks, such as comparing the traffic at different airports. Our transfer functions allow the users to focus on specific events of interest, without the distraction and clutter of the other air traffic.

3.6 bundling

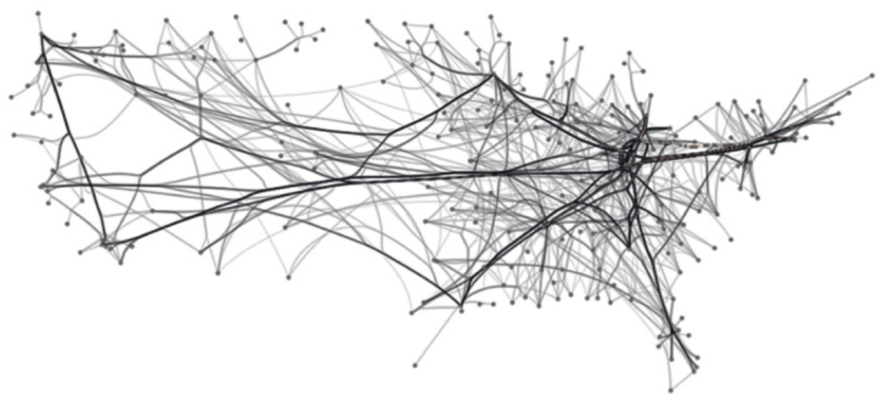
The previously shown techniques focus on multiple levels of detail at the temporal and spatial scale. However, we would like to have a different abstraction on the spatial scale, an abstraction that allows us to reason about groups of trails that are close in space and have similar start and end points. For this spatial abstraction, we will need a form of topological aggregation.

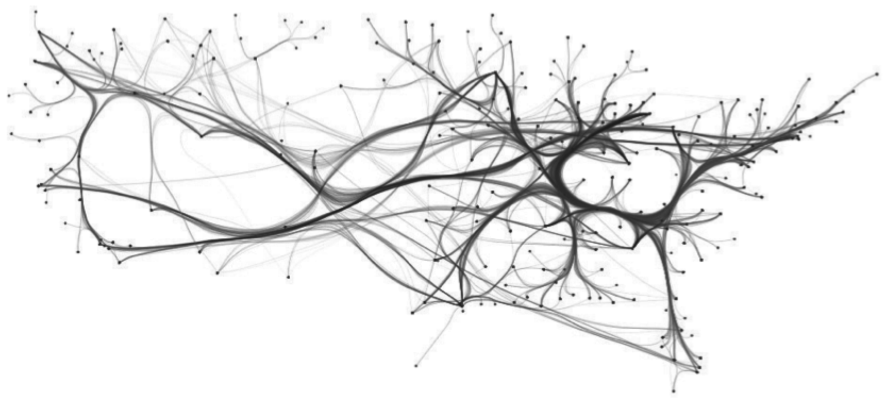
Currently, we can show the instantaneous positions of aircrafts and gradually increase Δ to increase the length of the trail segments and aggregation level. When we observe the nature of our trail data, we see that our trails closely follow a set of predefined routes. This effect can clearly be observed in Figure 14. We can exploit this behavior of trail data to create our topological abstraction. Trail segments that are parallel and closely spaced can be grouped together in order to simplify the visualization. There is a relatively recent technique that does precisely this: bundling. This makes bundling very well suited for these grouping tasks. While the output of these bundling algorithms does not preserve the route information, it can help us to better understand the *connectivity* between different airports [Lambert et al., 2010].

3.6.1 *Static Bundling*

Bundling techniques can be used to reduce clutter in the visualization of graphs by grouping spatially similar edges. These bundling techniques are capable of showing the coarse scale connectivity of large graphs. Many bundling algorithms have been proposed. Force-directed edge bundling (FDEB) creates bundles by attracting control points that are placed along the edges [Holten and van Wijk, 2009]. A Delaunay-based technique called geometry-based edge bundling (GBEB) is capable of creating curved edges [Cui et al., 2008]. Skeleton-based edge bundling (SBEB) uses a skeleton of the graph as bundling hints to create bundles with a lot of branches [Ersoy et al., 2011; Telea and Ersoy, 2010]. Kernel Density Estimation-based Edge Bundling (KDEEB) is one of the most recent and promising techniques, it is computationally efficient and the algorithm is very well suited to be implemented on a GPU. This parallelization makes it well suited to bundle very large graphs [Hurter et al., 2012].

Figure 16 gives an example of a possible outcome for different bundling techniques (FDEB, SBEB and KDEEB). It shows the result of applying bundling to a dataset containing movement data: flights above the USA. Bundling helps to show the coarse-scale flight patterns during a week.


(a) FDEB


(b) SBEB

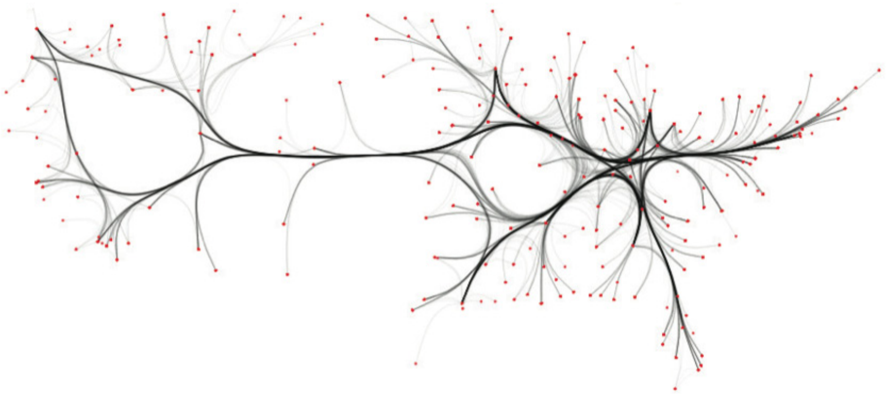

(c) KDEEB

Figure 16: Overview of different bundling techniques applied to a dataset containing flights above the USA (235 nodes, 2099 edges).

3.6.2 *Bundling for Dynamic Graphs*

Applying bundling to dynamic graphs introduces a new question: which part of we graph should we bundle? The bundling algorithm can be applied to the graph as a whole, or a (temporal) subset of the graph. For the latter case, Hurter et al. have recently extended the KDEEB algorithm to handle both streaming and sequence graphs [Hurter et al., 2013]. As explained in Section 3.2.3, our datasets have more resemblance with streaming graphs than with sequence graphs. Hurter et al. bundle streaming graphs by iteratively applying the bundle operation over a time window that slides through the time range of the graph. This offer a computational advantage, since the original KDEEB algorithm requires up to 10 iterations to bundle a single static graph. Furthermore, moving the time window for each iteration ensures smooth transitions within the bundled graph.

3.6.3 *Applying Bundling*

The KDEEB bundling algorithm has been successfully applied on aircraft routes by Hurter et al., but their implementation always shows the full trails and is not able to visualization additional attributes such as direction, altitude and velocity. We extend this method with our multivariate trail segment visualization. While recent implementations of this bundling algorithm are heavily parallelized and computationally efficient, it is still not feasible to bundle very large graphs for real time applications. Therefore, we chose to apply the KDEEB bundling algorithm on our data during a pre-processing step. We use our set of trails, without additional attributes, as the input of KDEEB and get a set of bundled trails as output. These bundled trails are then matched against the original trails, and the additional attributes are piecewise linearly interpolated over the length of the trails. Finally, we save both the original positions $\mathbf{po}_i$ and the bundled positions $\mathbf{pb}_i$ for each trail and use these, in combination with the additional attributes, as the input for our application. A user-defined bundle-factor W_b is used to linearly interpolate between the unbundled and bundled positions:

$$\mathbf{p}_i = W_b * \mathbf{pb}_i + (1 - W_b) * \mathbf{po}_i. \tag{7}$$

The final $\mathbf{p}_i$'s are visualized using the same techniques as described in Section 3.4.

Figure 17 shows the results of bundling our dataset above France for various values of W_b, all with $\Delta = 8$ hours. As can be seen, geographical information is lost in the bundling process, bundled trails indicate a connection between 2 airports, rather than the actual flying routes. Since we use the *original* attributes for color coding, using our directional color map still makes sense: the colors indicate the direc-

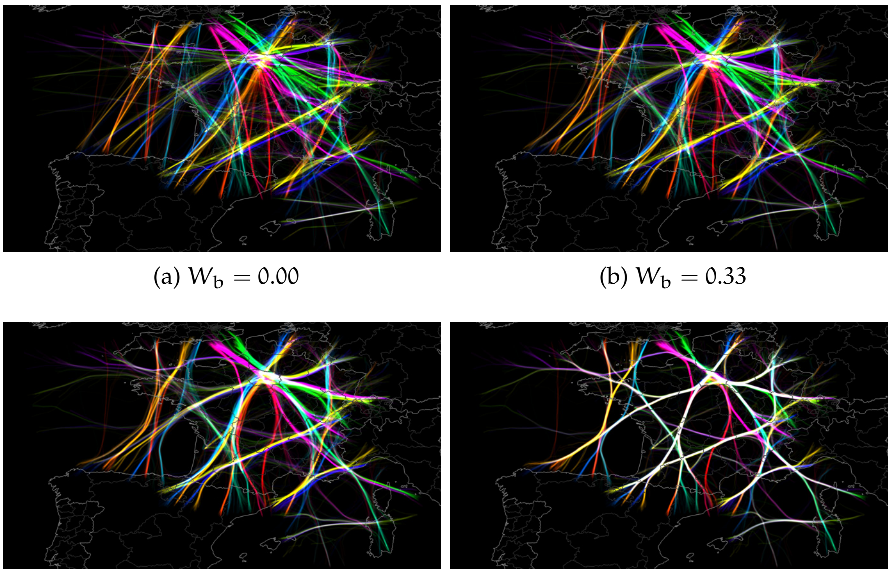

(a) $W_b = 0.00$
(b) $W_b = 0.33$
(c) $W_b = 0.66$
(d) $W_b = 1.00$

Figure 17: Bundling the French airspace dataset for various values of W_b

tion of the original unbundled paths, not the direction of the bundled paths. Paths that show up in white in Figure 17d contain flights that go in multiple (often opposite) directions. The bundled paths that contain a single color are the interesting finds, apparently most flights in these bundles are flying in roughly the same direction. This happens, for example, in the yellow/orange path in the west. Spotting this uniformity of direction in the unbundled variant (Figure 17a) is much harder. While the aggressive bundling of $W_b = 1.00$ can be useful in some cases, setting W_b to a somewhat lower value will often result in a more informative image. This can be seen in Figure 17c, that has $W_b = 0.66$. This image shows almost the same level of spatial abstraction as setting $W_b = 1.00$, while we can still distinguish parallel trails moving in opposite direction (e. g. the blue/yellow paths flying in northeast and southwest direction).

Figure 18 shows the result of applying our bundling step to a part of our worldwide dataset, where the height attribute is mapped to color. We can clearly see the main connections in this figure and are able to spot the different patterns concerning ascending and descending of the aircrafts around airports.

We can conclude that bundling can help us to create a topological abstraction of our flight data, and that while it does not preserve geological information, it does help us to understand connectivity between airports.

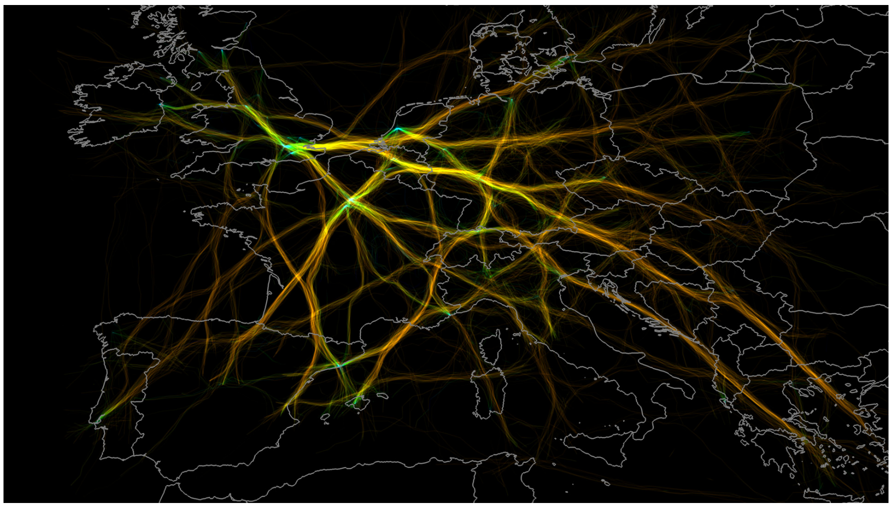


Figure 18: Bundling above a part of Europe, with height-based colormapping. Trail segments have a length of 2 hours.

3.7 CONGESTION DETECTION

The detection of congestion areas is an important task within movement data analysis [Scheepens et al., 2011; Hurter et al., 2009]. Congestion areas can be defined as spatial region with a high density of passing traffic within a certain time window. When the airspace gets congested, aircrafts have to wait for each other due to safety regulations, which results in delays and higher fuel consumptions. On a smaller spatial scale, highly congested zones have an increased risk of incidents and accidents. Reducing the amount of congestion zones can therefore result in reduced costs and risks, which makes it an interesting subject for ATC.

At the very basic level, we can use the density map from Equation 5 to detect zones with a high density of traffic. A similar technique was used by Scheepens et al. for ship data, that suggested a static density map for the entire studied time period. While this method is capable of showing areas where a lot of traffic passes during a large time period, this traffic does not necessarily pass within close temporal range. If two airplanes would collide on an otherwise sparsely populated area it would not be visible in this density map.

In order to circumvent these limitations we first construct a density map (similar to Equation 5) based on trail segments TC_2 within $w(t)$, which allows us to dynamically define the temporal scale for which we look for congestions.

The result of rendering this density map texture for our dataset of France can be seen in Figure 19. It allows the same transfer functions as our regular trail segment visualization, which enables us to focus the density map on the high altitude flights. For this result we chose $k_\alpha = 1.0$, since we are only interested in in-flight congestions, rather

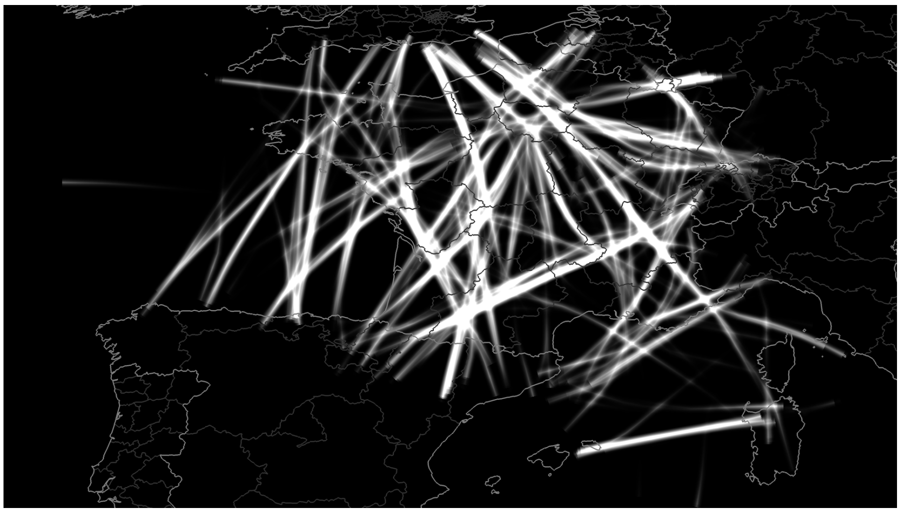


Figure 19: A 32 bit floating point precision density map is used for our congestion detection, it focuses on higher flights by setting $k_\alpha = 1.0$. $\Delta = 30$ minutes

than congestions on airstrips. While this Figure does indeed show density, it is very hard to visually find the congested areas. This happens because the areas with density > 1 are all visible as pure white; we can see no variation for the high density zones. At the same time, the dark gray areas are of no interest when looking for congested areas, while they clutter up the visual space.

In order to spot the truly congested areas more easily, we propose an extra rendering step after the generation of the density map. Instead of showing the original density ρ between 0 and 1, we render the congestion texture so that it shows the range between 1 and a user defined maximum $\rho_{\text{threshold}}$. We are only interested in $\rho > 1$ since, by definition, the density $\rho = 1$ at the instantaneous position of a single plane, and values of $\rho > 1$ indicate overlapping trail segments. This extra step is similar to the tone-mapping phase of HDR-rendering, that reduces the dynamic range to fit within the limited range of the medium that is used for display. In our case, the dynamic range of ρ (normally between 0 and ρ_{max}) is reduced to the range $[1, \rho_{\text{threshold}}]$

Figure 20 shows the results of reducing the range of Figure 19 to the range $[1, 5]$, and color mapping this with an alpha-based rainbow color map. In this case, we consider a zone to be minimally congested when 2 aircrafts pass the same location at 30 minutes from each other. Warmer colored and opaque areas show us the emerging congestion patterns quite clearly. Spotting these areas using any of the previously-used visualization is not a trivial task. We can see that most congestion areas line up with the major flight routes, which is not an unexpected find. However, we also see a few interesting small red areas on this map that do not follow the long structured routes.

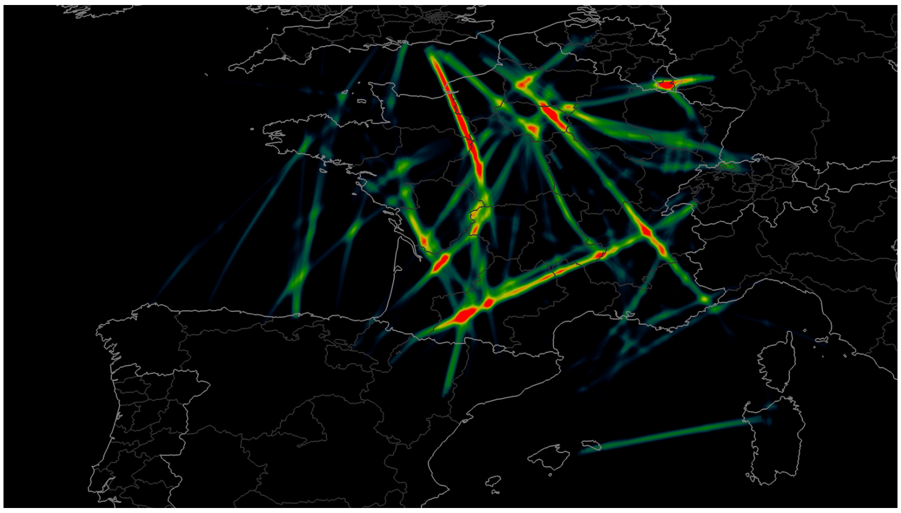


Figure 20: The final congestion detection map is constructed by a “tone mapping”-like step that reduces the 32 bits float texture to a 8 bit integer texture.

These congested areas must therefore be caused by intersecting trails that do not follow a single path.

3.8 CONCLUSION

This chapter has shown the various visualization techniques that we use in our application. Our main visualization techniques are density maps, animation and bundling. We can use our congestion detection visualization for more complex ATC tasks. Chapter 4 gives an in-depth explanation of the implementation of the techniques discussed in this chapter.

4 IMPLEMENTATION

In the previous chapter, we explained the theory behind our visualization techniques. In this chapter we give the important implementation details for some of these visualizations.

4.1 SHADER PIPELINE

In order to achieve interactive framerates for our previously defined visualizations, we have implemented the majority of our algorithms as OpenGL GLSL shaders. An overview of our shader pipeline is given in Figure 21, and is further explained below.

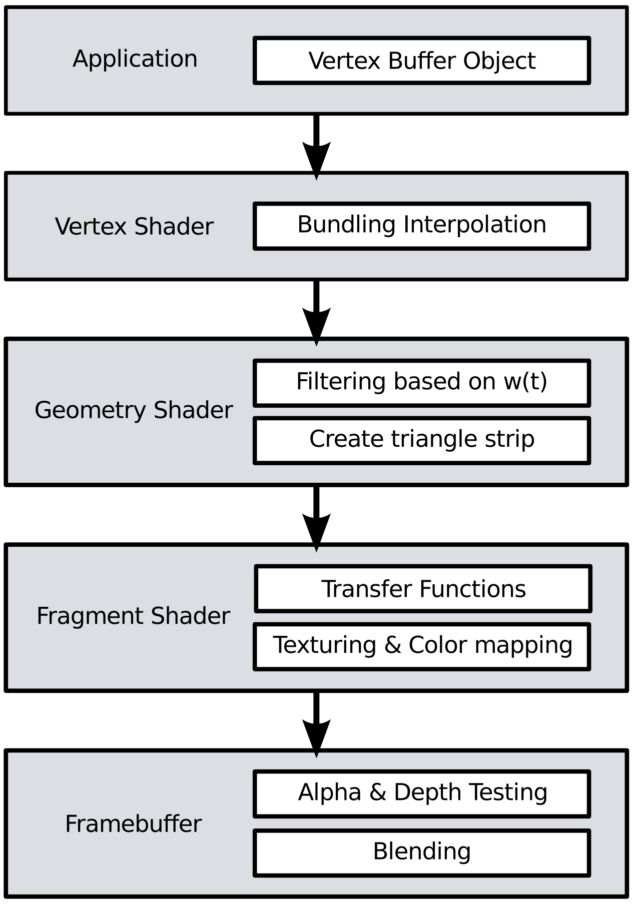


Figure 21: Shader Pipeline

All points $\mathbf{p}_i$ from TS are stored in a single Vertex Buffer Object (VBO) that is uploaded (and kept) on the video card. All points are stored in the same order as in TS and T, i. e. points are grouped per trail, and ordered by increasing values of t_i within these trails. The

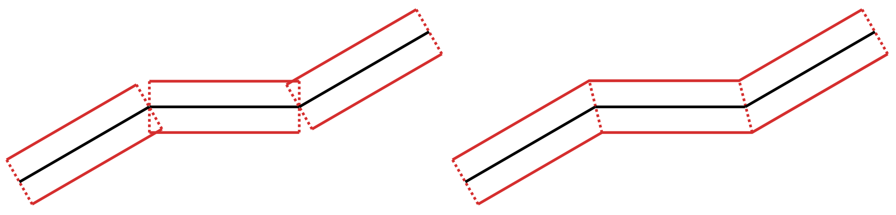

(a) Incorrect (b) Correct

Figure 22: Constructing a triangle strip from a line strip

resulting VBO contains one massive line strip, that will be split in separate lines further down the pipeline. The advantage of using one VBO, instead of a VBO per trail, is that there is no need to continuously switch the active VBO. When having a massive amount of trails, switching between all these VBO's becomes a considerable cost.

Our vertex shader does some standard transformations, and furthermore interpolates between the bundled and unbundled paths. This allows us to do the actual bundling offline, and only perform a very fast interpolation step in our application. The interpolation happens linearly based on a user-defined bundling factor.

Filtering of the points based on the sliding time window $w(t)$ to construct the collection of trails from Equation 6 happens in the geometry shader. The parameters t_{current} and Δ are passed as *uniforms* to the geometry shader, which only emits vertices for the points $\mathbf{p}_i$ that we need further in the pipeline (the fragment shader). This filtering technique, that is executed in parallel, drastically reduces the amount of data that needs to be processed by our fragment shader. Furthermore, our geometry shader splits the large line strip into triangle strips per trail. Doing the conversion from line strip to triangle strip on the fly reduces the size of our VBO. The geometry shader emits 2 vertices that are perpendicular to the original line segment. Setting the endpoints vertices parallel to the original endpoint of line strip will lead to incorrect results (see Figure 22a). Therefore, we need to align the vertices based on the neighboring vertices (see Figure 22b).

The execution of our transfer functions from Section 3.5 happens in the fragment shader. This means that our transfer functions are executed for every fragment (potential pixel), and not for every vertex. Applying the transfer functions on the vertices would be more computationally efficient, but would lead to incorrect results. This is due to the non-linear nature of our transfer functions, that would get linearly interpolated if they were applied on the vertex level. Color mapping the user-specified attribute and applying the right alpha texture based on δ also happens in the fragment shader.

4.2 CONGESTION DETECTION

We reuse our shader pipeline to generate the density map for our congestion detection visualization. However, instead of color mapping an attribute, we save the density values in in 32 bit floating point texture. This allows us to save the high-precision density values which we can further process.

The second step of our congestion detection, where we apply a tone-mapping step to reduce the dynamic range of our 32 bit texture, is implemented as a separate shader pipeline. The fragment shader in this pipeline is responsible for the tone-mapping and applying color mapping and texturing.

4.3 PERFORMANCE

Our application is implemented in Python with OpenGL visualizations, all performance critical parts are implemented as shaders. This allows us to render thousands of trails with millions of measurements in an interactive way. When using our largest dataset from Planefinder (14,711,646 sample points), we are able to maintain interactive framerates when we render up to 1 week of aggregated data. This is done on high-end consumer hardware (Intel Core i7 2.6 GHZ processor, 16 GB RAM, GeForce GTX 690 video card).

5 RESULTS

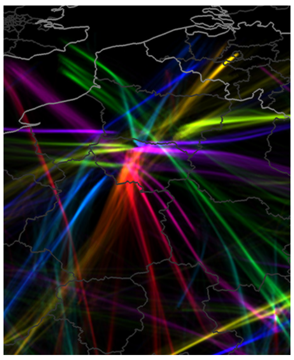
We use our visualization techniques to analyze two dataset (French airspace and Planefinder) for different time periods and locations. Our application is analyzed based on different ATC use cases. For each use case, we first introduce the use case, then discuss how our application handles this and outline the results and findings. The discussed use cases are generic, they are not evaluated with ATC controllers, but are based on previous experience with ATC/ATM workflows.

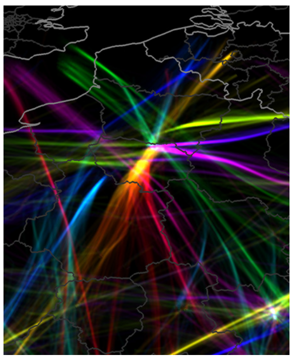
5.1 WEATHER INFLUENCES

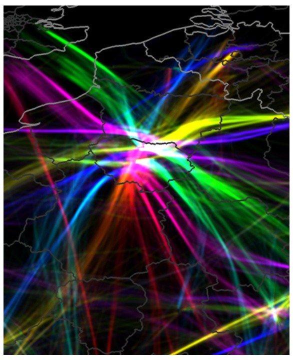
ATC operators and analysts are interested in analyzing the effects of weather, e. g. storms and strong wind currents, on air traffic [Hurter et al., 2013]. Therefore, we would like to ask the question: "how are observed traffic patterns correlated with wind currents?".

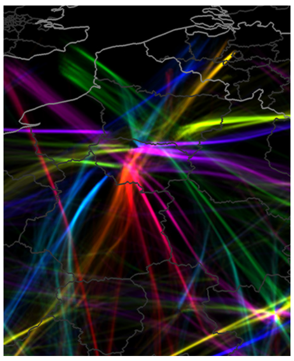
Figure 23 shows a cutout of our French airspace dataset for different days in the week, where each Figure spans a full day ($\Delta = 24$ hours). When watching the animation for these snapshots, one can easily spot the changes of patterns between the different days. The snapshots for Sunday and Monday (Figure 23a and Figure 23d) very clearly show different take-off and landing paths in comparison to Tuesday and Wednesday (Figure 23b and Figure 23d). In order to explain this interesting find, we looked up weather data for the given dates [Underground, 2013]. Figure 23g shows the wind direction for the time that is spanned by our French airspace dataset, which clearly indicates that Sunday and Monday had a different average wind direction (approximately West) than Tuesday and Wednesday (approximately North-East). This can explain the different take-off and landings paths, since air traffic controllers use wind as a factor when determining these paths. An other interesting phenomenon can be seen on Friday and Saturday: a vast increase in traffic, when compared to the other days. This is probably because an increased amount of holiday-related flights during the weekends.

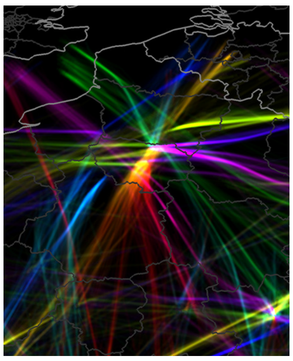
In order to verify our assumptions about the influence of wind direction on take-off and landing routes, we also examined our Planefinder dataset for similar patterns. When looking at Figure 24, we see 3 snapshots zoomed in on the Netherlands, all with a $\Delta = 24$ hours. We can see that Sunday (Figure 24a) and Monday (Figure 24b) had similar routes near the airport, and indeed, both days had a Northern wind direction. Thursday (Figure 24c) however, has different routes

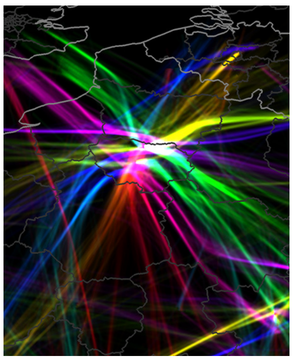

(a) Sunday 6th of April

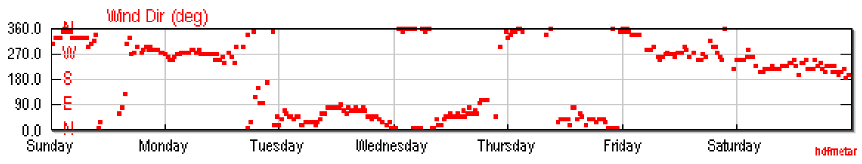

(b) Tuesday 8th of April


(c) Friday 11th of April


(d) Monday 7th of April

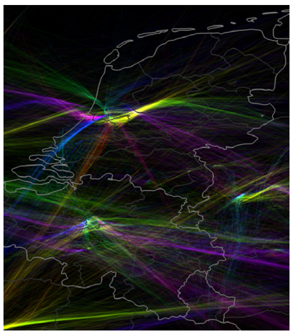

(e) Wednesday 9th of April


(f) Saturday 12th of April

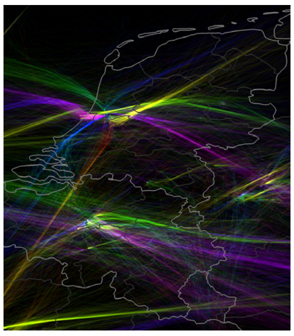


(g) Corresponding wind directions for the studied time period [Underground, 2013]

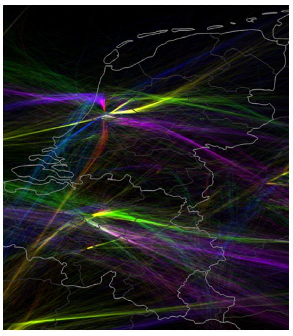
Figure 23: Direction-colored trails over a duration of 24 hours during April 2008 (all starting at 00:00) with an alpha-based emphasis on low flights (and airports). We see a clear difference in landing directions between Sunday/Tuesday and Monday/Wednesday. Friday and Saturday show a significant increase in traffic around Paris.



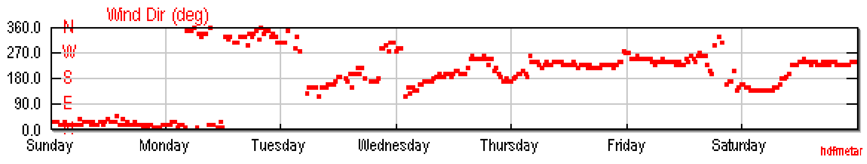
(a) Sunday 9th of June



(b) Monday 10th of June



(c) Thursday 13th of June



(d) Corresponding wind directions for the studied time period [Underground, 2013]

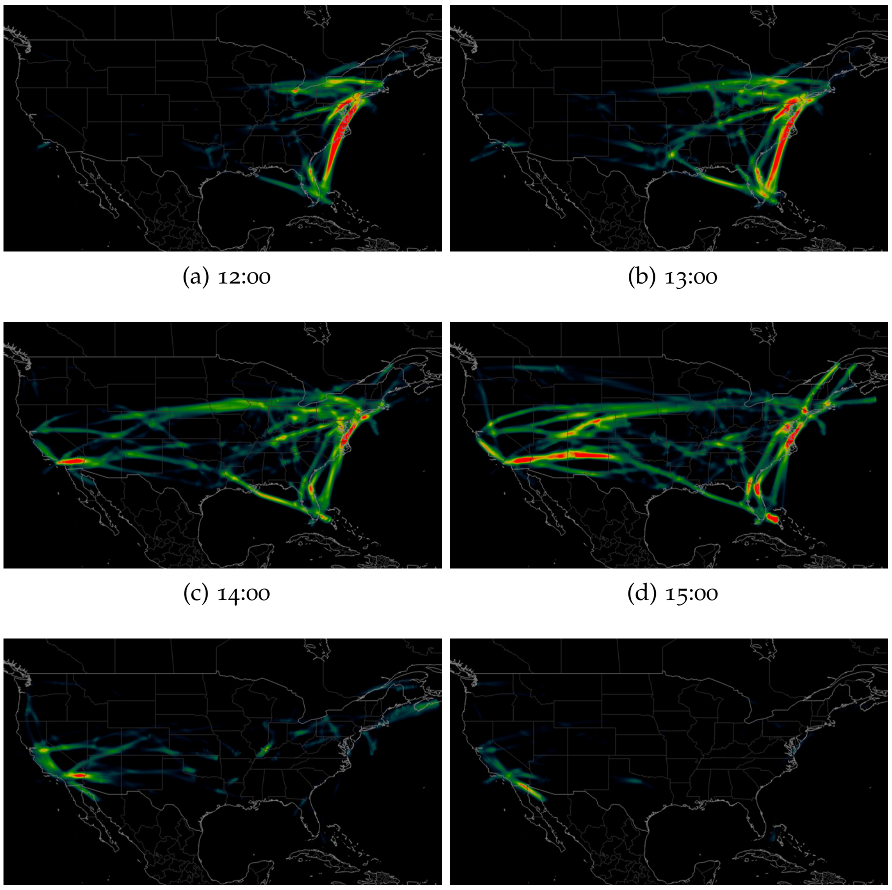
Figure 24: Direction-colored trails over a duration of 24 hours during June 2013 (all starting at 00:00) with an alpha-based emphasis on low flights (and airports), based on a zoom-in of the Netherlands in our Planefinder dataset. Thursday shows different landing and take-off routes when compared to Sunday and Monday, which can be correlated to the wind direction during this period.

around the airport, which correlates with the different (Southwestern) wind direction of that day. This confirms the finding in our previous dataset.

5.2 TIMEZONES

While our developed congestion map can be used to find zones with a high risk, it is also able to serve other uses. To optimize flying routes, air traffic controllers need to be aware of the congested zones for a given moment. Therefore, we ask ourself the question: "how does congestion vary over the day for a given territory?".

In Figure 25 we show the congestion map for different times of a single day above the USA, which clearly shows the different timezones. The first snapshot (Figure 25a) is taken at 12:00 UTC (07:00 Eastern Standard Time (EST), 04:00 Pacific Standard Time (PST)), and only shows intense traffic at the east coast. These are some incoming flights from Europe, flights that stay on the East coast, and a few flights on their way to the west. Gradually increasing the time until 15:00 UTC (10:00 EST, 07:00 PST), which can be seen in Figure 25b to Figure 25d, shows us an increase of traffic towards the west of the USA that stabilizes around 15:00 UTC. The last two snapshots (Figure 25e and Figure 25f) only show intense traffic near the west coast.


(a) 12:00

(b) 13:00

(c) 14:00

(d) 15:00

(e) 05:00

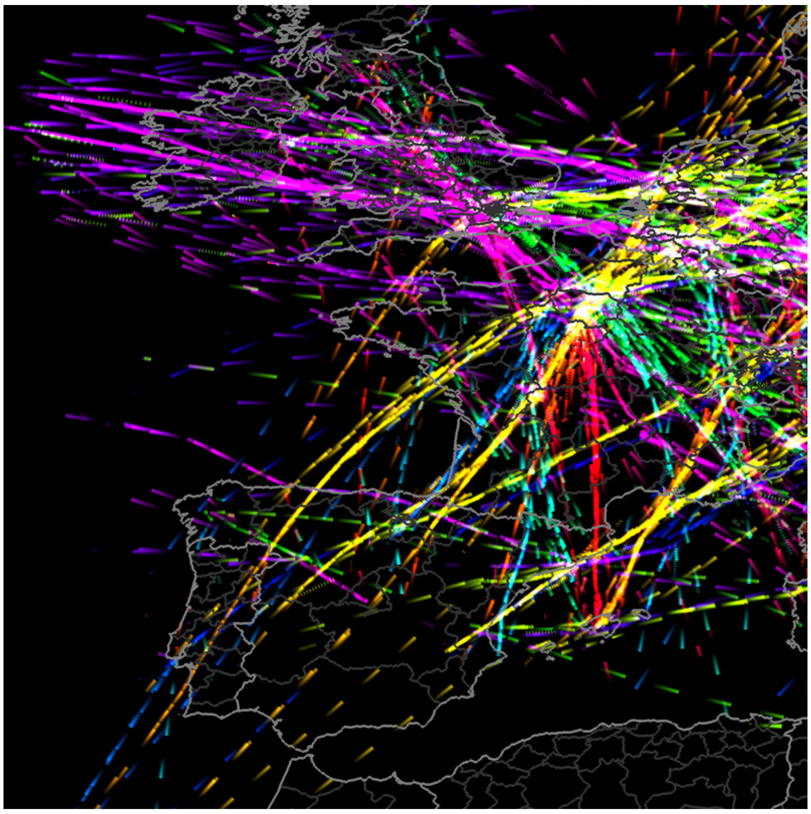
(f) 06:00

Figure 25: Congestion map for the USA, all times in UTC, intermediate times not shown for conciseness

5.3 DIRECTIONS ABOVE EUROPE

To evaluate our directional color map and arrow-like glyphs, we ask the question: “can we spot directional changes in highly dense areas?”

We demonstrate our directional colormap on a part of the Planefinder dataset in Figure 26. We see the same closely spaced parallel-but-opposite trails that we saw in our French airspace dataset. Furthermore, due to the directional colormapping and arrow-like shaped trails, we can easily spot temporal changes. For example, during the night, almost all flights that connect the USA to Europe are incoming flights from the USA (see Figure 26a). There is a time-zone difference of 5 hours between the east coast of the USA and the United Kingdom, and a flight from New York to London takes approximately 7 hours, which means that flights that land around 05:00 (local time) in London departed around 17:00 (local time) from New York, which is not a strange time to depart. However, a flight departing at 05:00 (local time) in London would probably not be the most popular choice. In contrast, when we look at Figure 26b, we see that at 14:00 UTC there are only outgoing flights to the USA, no incoming. This also makes sense, a plane landing at 14:00 (local time) in London would need to depart at 02:00 (local time) from New York.


(a) 05:00 UTC, incoming Flights from the USA

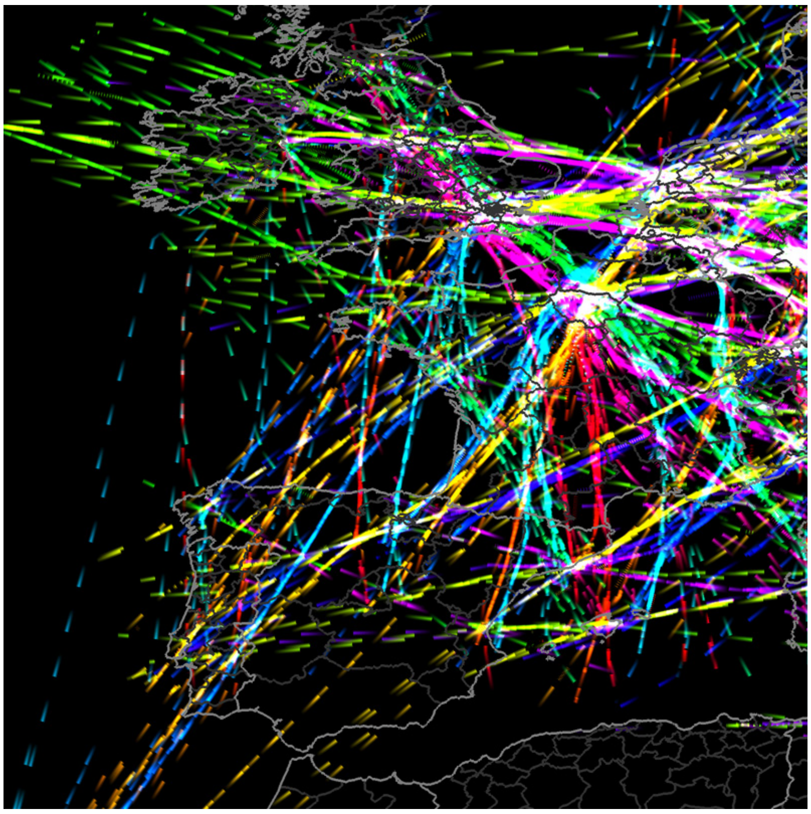
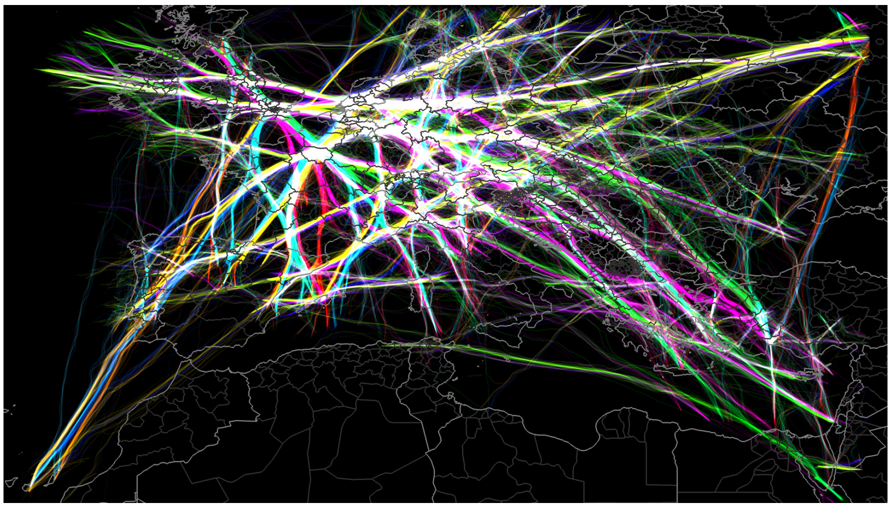

(b) 14:00 UTC, outgoing Flights to the USA

Figure 26: Flights above Europe captured by Planefinder. Trail segments are color coded by direction, and use $\Delta = 1$ hour and $\delta = 5$ minutes to emphasize directions over a longer time period with reduced clutter.

5.4 BUNDLING USING KDEEB

If there is a flight going from location A to location B, then a flight returning from B to A will probably also exist. It is interesting to analyze such pairs of linked flights. For example, we would like to know "how are flights in opposite directions balanced over a given time-and-space interval", and furthermore, we ask "are opposite and paired flights separated spatially, or do they follow the same route?".

In Figure 27 we show the Planefinder dataset above Europe, bundled using the KDEEB algorithm. A light bundling (Figure 27b) can still show the closely spaced opposite-but-parallel trails at various locations. However, when using strong bundling, most of these trails get merged. Due to our additive blending method and hue-based directional coloring, this results in white trails. Since Figure 27b contains mostly white trails, we can conclude that when looking at Europe for a longer time period ($\Delta = 8$ hours), most routes have nearly equal trail amounts in both directions. However, there are some interesting outliers to this general rule. For example, in France we can see a bright red/pink path going from Paris to the southeast, which mostly contains flights moving away from Paris. There are some other regions where similar effects occur, e. g. the blue path above Paris and the green and pink paths left of the United Kingdom.


(a) Lightly Bundled

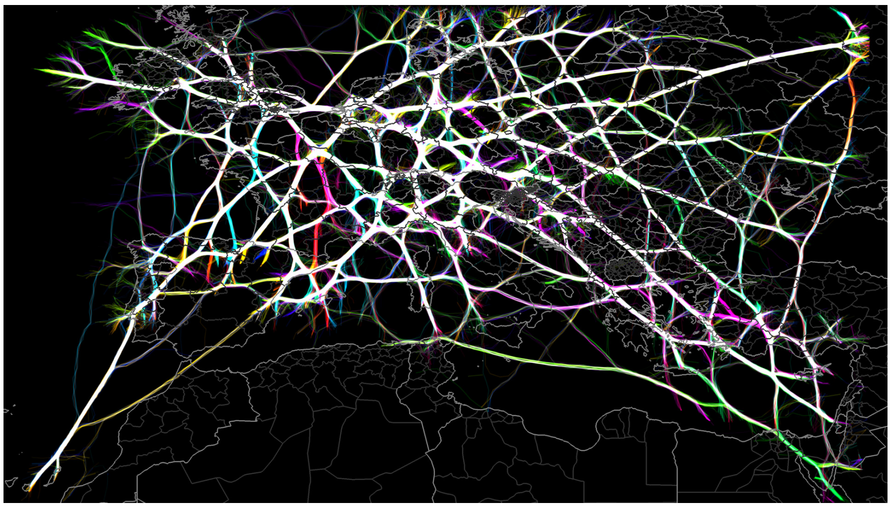

(b) Strongly Bundled

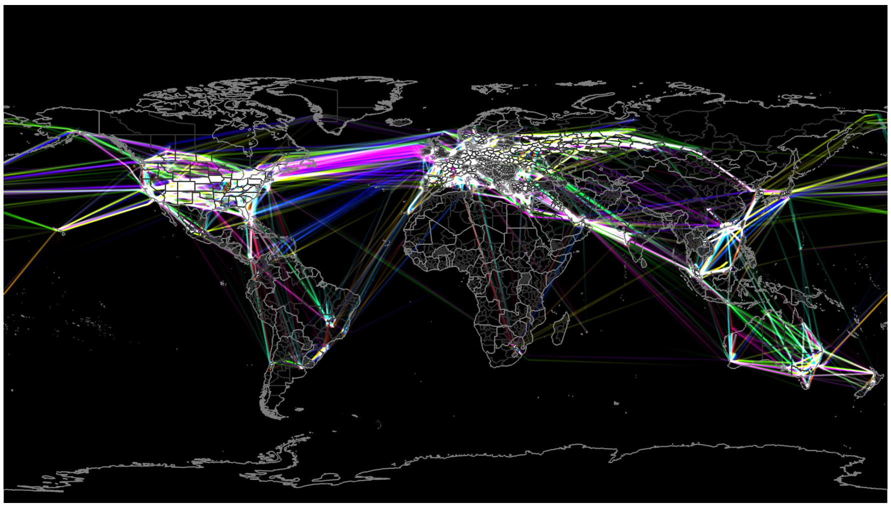
Figure 27: Bundled flights above Europe, color coded by their direction, $\Delta =$ 8 hours.

5.5 WORLDWIDE DATASET

Finally, we evaluate the scalability of our application. Here, our question is "how does our application scale visually and computationally for very large worldwide datasets".

When we zoom out even further, as we did in Figure 28, we can see the whole global dataset that is collected by Planefinder. When looking at Figure 28a we immediate spot a problem: due to the irregularities of global flights patterns, we have a few very dense zones that are almost completely white (e. g. Europe, USA), while it is hard to spot the traffic in the sparser areas. While it is impossible to see any details within Europe, this level of detail is still very useful. For example, we can distinguish the different lanes of traffic between Europe and the USA, and we can conclude that during this time window there is a lot more traffic going from Australia to Indonesia than the other way around.

We can significantly reduce the amount of clutter by decreasing Δ, as is demonstrated in Figure 28b. With these settings, we can identify individual airplanes at the sparse locations, while the dense zones are not too cluttered with white regions.


(a) Long trail segments, $\Delta = 16$ hours

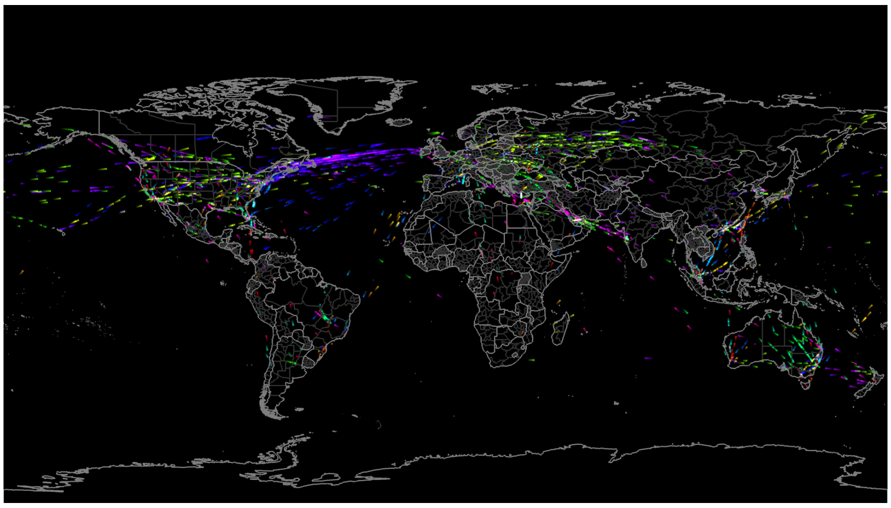

(b) Short trail segments, $\Delta = 20$ minutes

Figure 28: Worldwide dataset using directional color coding

6 CONCLUSION AND FUTURE WORK

There are many existing visualizations for air traffic data, some of which we have discussed in Section 2.1. However, most of these techniques focus on either showing instantaneous aircraft positions or aggregated data and often suffer from a lack of scalability and high amounts of clutter. We have presented a combination of visualization techniques that cover this gap. The introduction of a *sliding time window* allows us to continuously navigate between local detail (instantaneous positions) and global patterns (aggregation). Image-based visualization techniques based on density maps are used to show the amount of flights while maintaining visual scalability, animation is used to show the movement of aircrafts and change of flight patterns over time, and graph bundling is used show coarse-scale patterns between endpoints. Our glyph design uses shape, position and color to present several attributes at the same time. Computational scalability is achieved by our OpenGL shader based pipeline that is able to process many trails in parallel, this allows us to visualize thousands of trails with millions of data points at interactive framerates. We demonstrated our visualization techniques on several datasets ranging from hours over a single country to one month over the entire world.

6.1 LIMITATIONS

One of the limitations within this project had to do with our datasets. While our French airspace dataset is provided by the government, it is based on radar recordings, and not on modern fine grained techniques like ADS-B. This makes the French airspace dataset relatively coarse, both spatially and temporally, for the limited geographical space that is covers. On the other hand, our Planefinder dataset is based on ADS-B and consists of worldwide data, but this data is collected by ground stations that voluntarily submit their data to Planefinders website. As a result of this, rich and dense populated areas (e. g. Western Europe, coastal regions of the USA) have way higher coverage than poor and/or sparse populated areas (e. g. parts of Africa, Siberia). Furthermore, while ADS-B signals are broadcast every second, Planefinder only collects sample points for every 5 minutes. This means that our data could be up to 300 times more detailed in the temporal domain.

Next, although we were able to decrease the amount of overlapping flights in comparison to previous work [Hurter et al., 2009, 2013;

Krüger et al., 2013] by means of transfer functions, bundling and low values of δ, our application still suffers from a certain amount of visual clutter for dense flight areas viewed at a coarse scale or large values of Δ. There are solutions to decrease the amount of overlap even further, as is shown by Scheepens et al.. However, we choose to allow clutter in order to show individual outliers and enable continuous navigation between the different levels of detail. Running the application on a large high-resolution screen makes it easier to splot fine-grained patterns.

Finally, our application is limited in the amount of trail-attributes that can be visualized simultaneously. We currently exploit shape, length, position, and color in order to show 3 attributes at the same time. Showing more attributes is an open challenge to all similar research.

6.2 FUTURE WORK

While our method is more scalable both in visual space and computational complexity than current methods for the same types of datasets and analyses, our visualization still suffers from clutter due to the high amount of information that is shown. Improving the image-based visualization could help to reduce this clutter even further in the future. Our application could be further enhanced by adding interactive (visual) queries, e. g. lenses, that allow the user to compare and search spatio-temporal patterns of interest. For example, this would allow the user to search for flights departing from, or passing over a certain region.

Furthermore, it would be interesting to optimize our shader pipeline for even better temporal scalability. This would allow us to use even larger datasets and look at even coarser patterns, for example comparing different months of the year, or comparing the same month for different years.

To address the coverage issues of our Planefinder dataset, we could try to obtain ADS-B data from official authorities. Some governments have a nationwide array of base stations, and airlines collect all the data of their own airplanes.

Finally, our application can also be used for other movement data, such as the AIS-data that is broadcast by ships. As a small test, we collected 6 days of ship-data around the Netherlands (13,255 ships with 3,776,781 data points) from Shipfinder [Shipfinder, 2013], a website similar to Planefinder that is dedicated to the collection of AIS-data. Figure 29 shows a snapshot that is focused on the harbor of Rotterdam, with Δ set to 16 hours, and already gives us a nice impression of patterns of the nautical traffic. For instance, we can see an incoming (purple) and outgoing (green) lane for the harbor, and we can spot a continuously maneuvering pollution measuring vessel (the "Scheur-

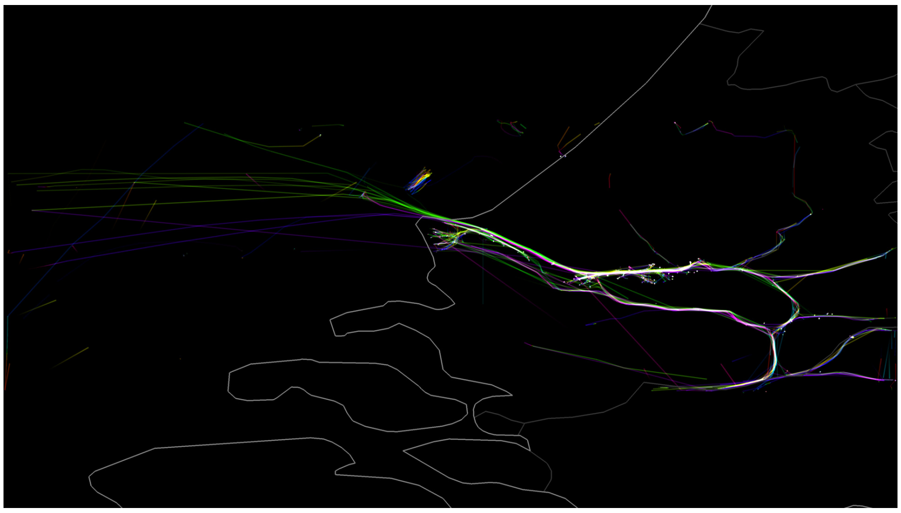


Figure 29: Visualizing ships with directional color coding, $\Delta = 16$ hours

rak") to the northwest of Rotterdam. However, in order to be useful for maritime traffic planners, our application will need to be adjusted to their specific requirements and use-cases.

BIBLIOGRAPHY

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

ADS-B. Automatic dependent surveillance broadcast, 2013. http://www.ads-b.com.

G. Andrienko, N. Andrienko, M. Burch, and D. Weiskopf. Visual analytics methodology for eye movement studies. *IEEE TVCG*, 18 (12):2889–2898, 2012. ISSN 1077-2626.

Gennady Andrienko, Natalia Andrienko, and Marco Heurich. An event-based conceptual model for context-aware movement analysis. *Int. J. Geogr. Inf. Sci.*, 25(9):1347–1370, September 2011. ISSN 1365-8816.

Natalia Andrienko and Gennady Andrienko. *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 3540259945.

Karl D Bilimoria, Banavar Sridhar, Gano B Chatterji, Kapil S Sheth, and SR Grabbe. Facet: Future atm concepts evaluation tool. *Air Traf c Control Quarterly*, 9(1), 2001.

NASA Ames Research Center. NASA Fact Sheet on FACET, 2010. http://www.aviationsystemsdivision.arc.nasa.gov/research/modeling/FACET_FactSheet_Final_20100920.pdf.

W. Cui, H. Zhou, H. Qu, P. Wong, and X. Li. Geometry-based edge clustering for graph visualization. *IEEE TVCG*, 14(6):1277–1284, 2008.

Department of Transportation. Automatic Dependent Surveillance-Broadcast (ADS-B) Out Performance Requirements To Support Air Traffic Control (ATC) Service; Final Rule. *Federal Register*, 75(103): 30160–30195, May 2010. URL http://www.gpo.gov/fdsys/pkg/FR-2010-05-28/pdf/2010-12645.pdf.

O. Ersoy, C. Hurter, F. Paulovich, G. Cantareira, and A. Telea. Skeleton-based edge bundles for graph visualization. *IEEE TVCG*, 17(2):2364 – 2373, 2011.

EU. laying down requirements for the performance and the interoperability of surveillance for the single european sky. *Of cial Journal of the European Union*, 305:35–52, November

2011. URL http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2011:305:0035:0052:EN:PDF.

Eurocontrol. NEST: Network Strategic Tool, 2013. www.eurocontrol.int/articles/airspace-modelling.

GAIN Group. Guide to methods and tools for airline flight safety analysis, 2004. flightsafety.org/files/analytical_methods_and_tools.pdf, 2nd ed.

H. Gaspard-Boulinc, Y. Jestin, and L. Fleury. Epoques: a user-centred approach to design tools and methods for atm safety occurences treatment. In *Proc. ESReDA (European Safety, Reliability and Data Association)*. European Commission, 2003.

D. Holten and J. J. van Wijk. Force-directed edge bundling for graph visualization. *CGF*, 28(3):670–677, 2009.

C. Hurter, B. Tissoires, and S. Conversy. FromDaDy: Spreading data across views to support iterative exploration of aircraft trajectories. *IEEE TVCG*, 15(6):1017–1024, 2009.

C. Hurter, O. Ersoy, and A. Telea. Moleview: An attribute and structure-based semantic lens for large element-based plots. *IEEE TVCG*, 17(12):2600–2609, 2011.

C. Hurter, O. Ersoy, and A. Telea. Graph bundling by kernel density estimation. *CGF*, 31(3):435–443, 2012.

C. Hurter, O. Ersoy, and A. Telea. Smooth bundling of large streaming and sequence graphs. In *Proc. Paci cVis*, pages 374–382. IEEE, 2013.

ITU. Technical characteristics for an automatic identification system using time division multiple access in the vhf maritime mobile band., 2001.

R. Krüger, D. Thom, M. Wörner, H. Bosch, and T. Ertl. TrajectoryLenses - a set-based filtering and exploration technique for long-term trajectory data. *CGF*, 32(3):451–460, 2013.

A. Lambert, R. Bourqui, and D. Auber. Winding roads: routing edges into bundles. In *Proceedings of the 12th Eurographics / IEEE - VGTC conference on Visualization*, EuroVis'10, pages 853–862, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association. doi: 10.1111/j.1467-8659.2009.01700.x. URL http://dx.doi.org/10.1111/j.1467-8659.2009.01700.x.

O.D. Lampe and H. Hauser. Interactive visualization of streaming data with kernel density estimation. In *Paci c Visualization Symposium (Paci cVis), 2011 IEEE*, pages 171–178, 2011. doi: 10.1109/PACIFICVIS.2011.5742387.

Ove Daae Lampe, Johannes Kehrer, and Helwig Hauser. Visual analysis of multivariate movement data using interactive difference views. In *VMV*, pages 315–322, 2010.

C. Letondal, C. Hurter, R. Lesbordes, J. L. Vinot, and S. Conversy. Flights in my hands: coherence concerns in designing strip'tic, a tangible space for air traffic controllers. In *Proc. ACM CHI*, pages 2175–2184, 2013.

He Liu, Yuan Gao, Lu Lu, Siyuan Liu, Huamin Qu, and L.M. Ni. Visual analysis of route diversity. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 171–180, 2011. doi: 10.1109/VAST.2011.6102455.

A. Malik, R. Maciejewski, B. Maule, and D.S. Ebert. A visual analytics process for maritime resource allocation and risk assessment. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 221–230, 2011. doi: 10.1109/VAST.2011.6102460.

A. Marzuoli, C. Hurter, and E. Feron. Data visualization techniques for airspace flow modeling. In *Proc. Intelligent Data Understanding (CIDU)*, pages 79–86, 2012.

Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):pp. 1065–1076, 1962. ISSN 00034851. URL http://www.jstor.org/stable/2237880.

PlaneFinder. Live flight status tracker, 2013. http://planefinder.net/.

Maria Riveiro and Göran Falkman. The role of visualization and interaction in maritime anomaly detection. In *IS&T/SPIE Electronic Imaging*, pages 78680M–78680M. International Society for Optics and Photonics, 2011.

Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3): 832–837, September 1956. ISSN 0003-4851. doi: 10.1214/aoms/1177728190. URL http://dx.doi.org/10.1214/aoms/1177728190.

R. Scheepens, N. Willems, H. van de Wetering, G. Andrienko, N. Andrienko, and J. J. van Wijk. Composite density maps for multivariate trajectories. *IEEE TVCG*, 17(12):2518–2527, 2011.

R. Scheepens, N. Willems, H. Van de Wetering, and J.J. van Wijk. Interactive density maps for moving objects. *Computer Graphics and Applications, IEEE*, 32(1):56–66, 2012. ISSN 0272-1716. doi: 10.1109/MCG.2011.88.

Shipfinder. Live ship status tracker, 2013. http://shipfinder.co/.

B. W. Silverman. *Density estimation: for statistics and data analysis*. London, 1986.

A. Telea and O. Ersoy. Image-based edge bundles: Simplified visualization of large graphs. *CGF*, 29(3):543–551, 2010.

Thales, Inc. CoFlight Flight Data Processing Framework, 2013. `www.thalesgroup.com`.

Weather Underground. Historical weather data, 2013. `http://www.wunderground.com/history/`.

Niels Willems, Huub van de Wetering, and Jarke J. van Wijk. Visualization of vessel movements. In *Proceedings of the 11th Eurographics / IEEE - VGTC conference on Visualization*, EuroVis'09, pages 959–966, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association. doi: 10.1111/j.1467-8659.2009.01440.x. URL `http://dx.doi.org/10.1111/j.1467-8659.2009.01440.x`.

Niels Willems, Huub van de Wetering, and Jarke J. van Wijk. Evaluation of the visibility of vessel movement features in trajectory visualizations. In *Proceedings of the 13th Eurographics / IEEE - VGTC conference on Visualization*, EuroVis'11, pages 801–810, Aire-la-Ville, Switzerland, Switzerland, 2011. Eurographics Association. doi: 10.1111/j.1467-8659.2011.01929.x. URL `http://dx.doi.org/10.1111/j.1467-8659.2011.01929.x`.