Interactive image-based information visualization for aircraft trajectory analysis

C. Hurter
ENAC/University of Toulouse, France
christophe.hurter@enac.fr

S. Conversy
ENAC/University of Toulouse, France
stephane.conversy@enac.fr

D. Gianazza
ENAC/University of Toulouse, France
gianazza@recherche.enac.fr

A. C. Telea
University of Groningen, the Netherlands
a.c.telea@rug.nl

Abstract—
The Air Traffic Control (ATC) domain is an example of the Big Data challenges. Data is formed by sets of airplane trajectories, or trails, which in turn record the positions of an aircraft in a given airspace at several time instants, and additional information such as flight height, speed, fuel consumption, and metadata (e.g. flight ID). Analyzing and understanding time-dependent data poses several non-trivial challenges to information visualization. In this paper, we present a set of novel methods to analyze aircraft trajectories with interactive image-based information visualization techniques. We address the scalability challenges in terms of data manipulation and open questions by presenting a set of related visual analysis methods that focus on decision-support in the ATC domain. All methods use image-based techniques, in order to outline the advantages of such techniques in our application context, and illustrated by means of use-cases from the ATC domain. For each use-case, we outline the type of questions posed by domain experts, data involved in addressing these questions, and describe the specific image-based techniques we used to address these questions. For each such technique, we describe the visual representation and interaction mechanisms that have been used to address its goals. We illustrate these use-cases with real-life datasets from the ATC domain.

1 INTRODUCTION

Our society has entered a data-driven era, in which not only enormous amounts of data are being generated every day [50], but there are also growing expectations placed on the analysis of these data.

Movement data is an active research area which generates increasingly large Big Data collections [2, 1, 11] with impact in the fields of security (e.g. crisis analysis), economics (e.g. aircraft fuel consumption, human migration patterns) and healthcare (e.g. epidemics propagation). Within this domain, the Air Traffic Control (ATC) field is a good example of the Big Data challenges. Data is formed by sets of airplane trajectories, or trails, which in turn record the positions of an aircraft in a given airspace at several time instants, and additional information such as flight height, speed, fuel consumption, and metadata (e.g. flight ID). Movement data can be automatically collected, e.g. using GPS, radar, or RFID sensing technologies.

ATC users regularly analyze traffic to address the following tasks:

- understand past conflicts and then improve safety with adequate evolutions;
- assess new onboard and ground safety systems and the resulting aircraft trails;
- devise new air space organization and procedures to handle traffic increase;
- compare trails with environmental considerations (fuel consumption, noise pollution, vertical profile comparison);
- study profitability from a business viewpoint (number of aircraft on a specific flight route per day, number of aircraft that actually landed at a specific airport),
- examine specific flight trajectories for ATC training purposes.

Analyzing and understanding time-dependent data poses several non-trivial challenges to information visualization, as follows:

Scalability: Movement datasets are by their very nature several orders of magnitude larger than static datasets. Fig. 1 a illustrates this: A single day of air traffic over France contains some 20000 trajectories (shown here color-coded by height), each having hundreds or even thousands of data points. Such datasets are orders of magnitude larger for longer time periods [30]. This poses both *computational* problems (how to quickly detect patterns of interest in dynamic, streaming, data) and *representation* problems (how to present such patterns to the user in an understandable manner). The representation problem is particularly hard. First, there is limited 2D screen real-estate to display large amounts of data. This problem becomes even harder as the amount of attributes per data point increases. Secondly, displaying temporal (dynamic) patterns is considerably harder than showing static (e.g. purely spatial) patterns.

Query response time: In ATC, the ability to make timely decisions based on available data is crucial to deal with crisis situations, e.g. the case when one has to dynamically re-route part of an air traffic schedule to handle emergencies created by aircraft malfunction, extreme weather, or unexpected ground delays. Doing this in near-real-time requires easy-to-use ways to query and assess the changing situation presented by ATC data and also to input decisions interactively. This is challenging, since movement datasets have become simply too large and often have too short a lifespan, i.e. change too rapidly, for classical visualization or analysis methods to be able to handle it properly. For example, in ATC

users must perform dynamic requests (response time < 100 ms) on movement datasets containing over 1 million data points. Formulating such queries in a declarative, text-based manner, such as using SQL, is ineffective. It is very difficult to specify features like 'select trajectories where this part of the trajectory is straight' or 'where this part has a constant climbing rate. Even if a pre-defined set of queries were available to cover all expected use-cases, posing such queries via a classical text or menu-based interface would be too slow.

Data quality: In addition to the above problems, ATC controllers have to deal with datasets that contain many errors and uncertainties. ATC recording is done in a periodic manner (in our database: a radar plot per aircraft every 4 minutes). However, a plot can be missed, or have erroneous values because of physical problems that occurred at the time of recording. The challenge is how to design visual analysis algorithms that (a) are robust against such uncertain or missing data and (b) expose data quality problems so users understand whether and how this influences their decisions.

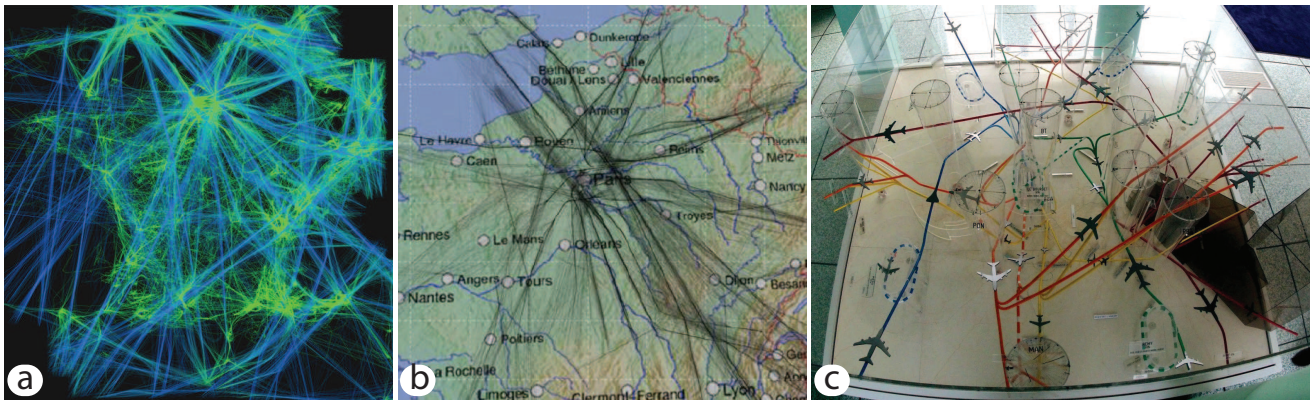

Fig. 1. (a) All flights over the French territory over a single day (July 5^{th} 2006). (b) Combined geographical and flight data visualization over the Paris region. (c) Abstract model of aircraft trajectories, displayed in the ATC tower at Roissy, Paris, France.

1.1 Image-based visual analytics

To address scalability, visual analytics research in recent years has increasingly focused on the development of a particular family of methods: Image-based visualization. Also known as dense-pixel or space-filling visualizations, such methods essentially use every screen pixel to display information, thereby addressing visual scalability. Given recent developments in general-purpose graphics processing unit (GPGPU) computing, such algorithms can be effectively implementing using the highly-parallel computational and graphics engines available on modern graphics card, thereby addressing computational scalability.

However, effective and efficient usage of such techniques to address decision-support in the ATC domain still remains challenging. Specific open questions include the following:

- How to efficiently implement interactive query techniques atop of image-based techniques in order to address the low-response-time and ease-of-use requirements of the ATC domain?
- Which visual metaphors efficiently scale to display large amounts of time-dependent multivariate movement data with minimal clutter and maximal effectiveness?
- How to validate such novel techniques based on real-world ATC datasets and tasks?

In this paper, we address the above challenges and open questions by presenting a set of related visual analysis methods that focus on decision-support in the ATC domain. All methods use image-based techniques, in order to outline the advantages of such techniques in our application context, and illustrated by means of use-cases from the ATC domain. For each use-case, we outline the type of questions posed by domain experts, data involved in addressing these questions, and describe the specific image-based techniques we used to address these questions. For each such technique, we describe the visual representation and interaction mechanisms that have been used to address its goals. We illustrate these use-cases with real-life datasets from the ATC domain.

The structure of this paper is as follows. Section 2 outlines related work with a focus on image-based visualization methods for the ATC domain. Sections 3-6 present four types of image-based visualization techniques and their applications in ATC data analysis: interaction, density maps, graphs, and animation. Section 7 discusses the observed results from the usage of image-based visual analytics for the ATC domain. Section 8 concludes the paper.

2 Related Work

Related work can be structured along several directions: commercial-grade ATC systems (Sec. 2.1), visualization techniques (Sec. 2.3), and data analysis techniques (Sec. 2.2). Additional related work is introduced later in the paper in the context of specific problems.

2.1 ATC systems

Several interactive visual systems are well-known in the current ATC practice. The Future ATM Concepts Evaluation Tool (FACET) [47] is a flexible software tool that is capable of quickly generating and analyzing thousands of aircraft trajectories. It provides users with a simulation environment using aircraft performance profiles, airspace models, weather data, flight schedules, and the tool models trajectories for the climb, cruise, and descent phases of flight for each type of aircraft. A graphical interface displays traffic patterns in two and three dimensions, under various current and projected conditions for specific airspace regions. Similar systems have been developed by Eurocontrol, the European Organization for the Safety of Air Navigation. For example, the Network Strategic Tool (NEST) [17] is a stand-alone desktop application used by air traffic practitioners for airspace structure design and development, capacity planning and post-operations analysis, the organization of

traffic flows, the preparation of scenarios for fast time simulations, and ad-hoc studies at local and network level. EPOQUES [23] is a specific system which aims to gather and analyze radar recordings and audio communications. It proposes underlying tools and methods to treat Air Traffic Management (ATM) safety occurrences, such as helping operators to detect and analyze situations when two aircraft went beyond safety distance. Next-generation systems, currently under development at the writing of the present article, include CoFlight, a flight data processing (FDP) open-architecture framework for the storage, analysis, and visualization of 4D (spatio-temporal) flight data [49]. A comprehensive list of over 50 ATC-related systems and tools is given in [20].

Although all above authors and organizations stress the importance of visualization for the effective and efficient assessment of situations and events in ATC use-cases, visual techniques currently used in ATC tools are relatively limited. In particular, visual scalability and the ease of rapidly posing complex queries are still open challenges.

2.2 Analysis techniques

The possibility of extracting spatial events from different types of spatio-temporal data is discussed in [3]. However, few concrete methods are presented. Moving object databases compute various dynamic attributes (numeric functions and predicates) and provide a powerful query language for extracting parts of trajectories based on these attributes [24]. Andrienko *et al.* describe a visual query tool that computes and visualizes some of the movement characteristics and distances to selected context elements and filters trajectory segments based on values of one or more dynamic attributes [2]. A good overview of movement pattern analysis is given in [36].

However, none of the previous algorithm or software uses the GPGPU technique or a pixel-based algorithm.

2.3 Visualization techniques

Pixel-based techniques are increasingly popular techniques for visualizing large amounts of data because they are able to display large data sets at a high resolution [38]. Keim created the basis of displaying large data sets at a high resolution to support the analysis of huge amounts of data [33]. Fekete and Plaisant pioneered scalable pixel-based techniques for information visualization (Infovis) [18]. They showed how each screen pixel can be used to show a data item, thereby displaying millions of data elements, and proposed interactive queries for scatterplots and treemaps.

Interaction and representation with large data heavily rely on algorithms to compute and draw the representation, and algorithms to transform the user input into view changes or data queries. Not only do the performance of these algorithms determine what representations can be used in practice, their nature also has a strong influence on what the visualizations look like. The algorithms that are used classically in InfoVis are expressed in the data space (e.g. computation on geographic locations). We consider an alternative approach: algorithms expressed in the graphic space (image-based algorithms). This consists of two steps: first, a data representation is built using straightforward InfoVis techniques; second, the resulting image undergoes purely graphical transformations using image processing techniques.

Image-based techniques (IBT) leverage changes in the computer graphics pipeline: Recent graphics cards are designed to exploit their computing power to perform massively parallel computations by so-called GPGPU techniques. For instance, Scheepens *et al.* use density maps to display thousands of trajectories of nautical vessels on 2D maps and emphasize congestion areas that cause navigation problems [45]. Lambert *et al.* leverage GPGPU techniques to quickly compute 2D uncluttered layouts of large aircraft trajectories [35].

Image-based Infovis techniques essentially combine the *visual* scalability of pixel-based techniques with the *computational* scalability of image-based techniques. Our approach, that we call image-based InfoVis, differs from these works and most other Infovis works in that we plan not only to use pixel-based visualization, but also to perform data exploration using image-based algorithms. Classical image processing techniques such as sampling and filtering can be used to construct continuous multiscale representations, which further enhance scalability. By synthesizing color, shading, and texture at a pixel level, we achieve a much higher freedom in constructing a wide variety of representations that is able to depict the rich data patterns we aim to analyze. A good overview of GPGPU techniques for image-based visualization is given in [40].

Various combinations of pixel-based and image-based techniques have been specifically applied to explore movement data. Interactive *brushing-and-linking* can be used to examine large multivariate trajectory datasets, by selecting up to three dimensions to be displayed in a single 2D view [32]. Brushing means selecting a subset of interest from the visible data items with an input device, usually the mouse. Selection can be performed using nearest-point or region (rectangular, circular, or free-form) techniques. Data items whose visible representations fall inside the selection area are highlighted. Next, additional information is displayed for the highlighted items, either in-place in the view where brushing was done, or in other views. Brushing is typically used in connection with linking. In this context, when a data element is selected in one view, all other visual representations of the same data element in other views are automatically highlighted. This allows following the relative position of the brushed data elements in several views.

Density maps are an effective instrument to tackle the visual scalability problem, by aggregating spatially close information for trajectory analysis [2, 1, 39]. Multimodal interactions help users in posing complex queries with little effort [31, 37]. Bundling techniques are effective in showing the coarse-scale connectivity structure of a set of trails that link a set of spatial locations in a clutter-free manner [29, 27, 16, 35, 8]. Bundling computational scalability is significantly boosted by using image-based techniques for both static trajectory datasets [29] and time-dependent, or streaming, trajectories [30]. Focus+context interaction techniques help in further reducing clutter and posing complex spatial-and-data queries in bundled visualizations [28]. Such techniques have also been extended to other data spaces, such as image exploration [6].

In the remainder of this paper, we present several applications of image-based visual analytics techniques for the exploration of ATC datasets. We structure our exposition based on the types of techniques being used: Section 3 focuses on interaction techniques. Section 4 focuses on density maps. Section 5 focuses on graph-based techniques. Finally, Section 6 presents the usage of animation.

3 Interaction techniques

We have developed FromDaDy (which stands for 'FROM DAta to DisplaY'), a visualization tool that tackles the challenge of representing, and interacting with, numerous aircraft trajectories involving uncertainties. FromDaDy employs a simple paradigm to explore multidimensional data based on scatterplots, brushing, pick and drop, juxtaposed views and rapid visual configuration. The fundamental new aspect of FromDaDy compared to existing ATC visualization systems, is to enable users to spread data across several linked *views* using interactive techniques. This enables users to filter, remove, and add trajectories iteratively, and examine results from different viewpoints, thus supporting the execution of complex queries. Below, we present several air traffic analyses supported by the interaction techniques in FromDaDy.

3.1 Aircraft trajectory data exploration

Figure 2 illustrates the base functionality of FromDaDy. Image (a) shows the user interface. Here, users can 'connect' the available input data variables to visual attributes to build custom analyses. To construct image (b), for example, we map the dataset's longitude, latitude, and flight

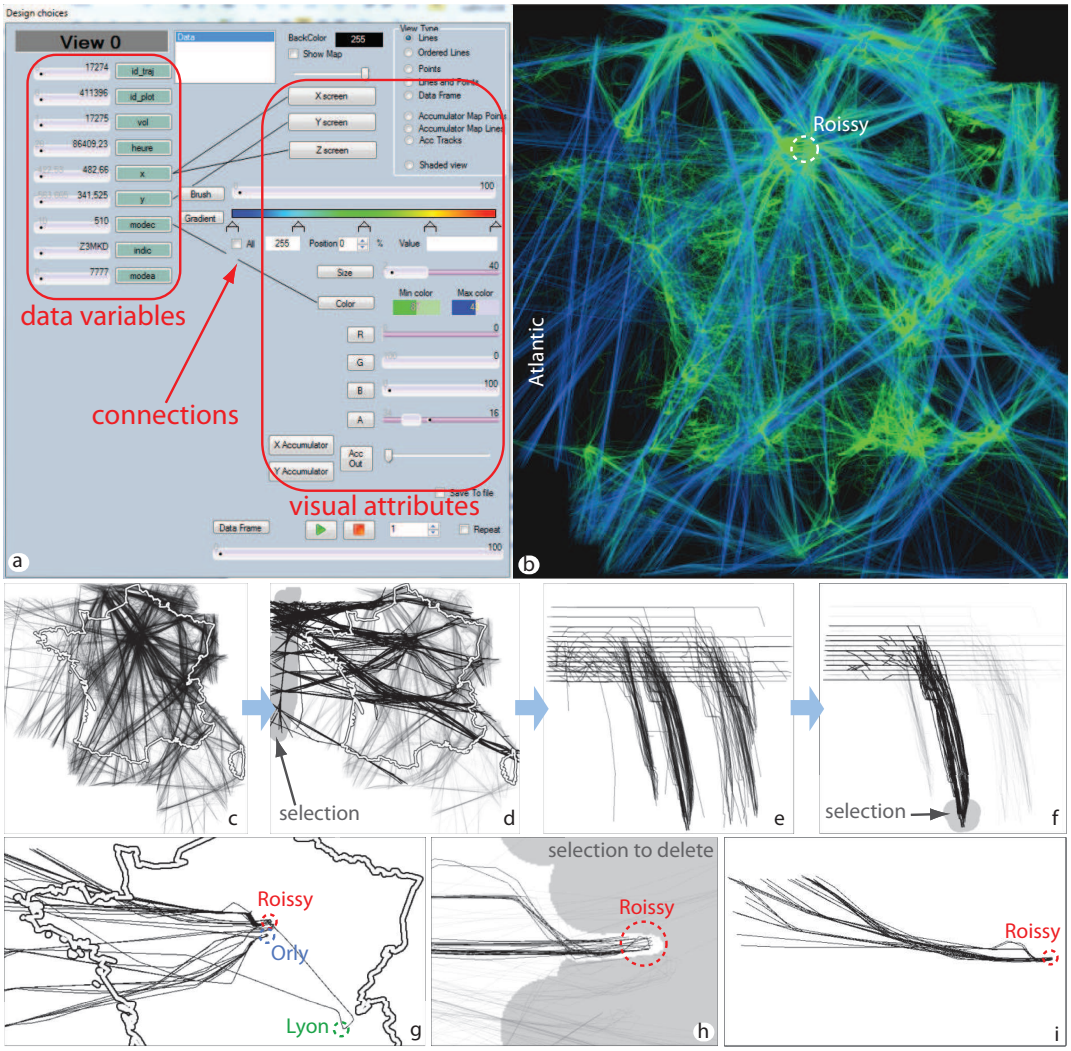

Fig. 2. Aircraft trails visualization in FROMDADY (Sec. 3.1). (a) Visualization configuration interface. (b) Trails colored by height. (c-f) Interactive querying.

height to the screen's x, y and color (using a blue-to-red colormap). Flight trails are rendered using alpha blending. Hence, brightly colored areas correspond to zones densely covered by flights, such as airports. The emerging image (b) shows two different patterns: Long straight blue lines are high-altitude flights. Below, we see an intricate mesh of short green lines (low altitude flights). This correlates well with the aircraft type: High altitude flights are typically international flights which connect the French territory with other countries. Low-altitude flights are light aircraft, which is mainly confined to the French mainland, due to visibility reasons. The Atlantic French coastline is clearly visible here as the left border of the green pattern.

3.1.1 Simple queries

To illustrate FROMDADY's iterative exploration, consider the following query: How to find in Fig. 2 (b) transatlantic aircraft that landed or took off at the Roissy airport during one day? To do this, we perform the following interactive selection steps. First, we brush the left area of the visualization to select aircraft that flew over the ocean (image (d)). Next, we change the configuration to a vertical view (x=altitude, y=latitude, image (e)). Horizontal lines in this view indicate different flying levels. Next, we brush to select aircraft that have a very low altitude at the airport's longitude, *i.e.*, are landing or taking off (image (f)). Finally, we revert the view configuration to a top view (x=longitude, y=latitude). Upon closer inspection of the resulting (image (g)), we see that we have selected flights ending to *two* close airports, Roissy and Orly, as these have nearly the same longitude. We also discover an interesting outlier – a transatlantic flight that arrives at Roissy via an unexpected transit through Lyon. To refine the selection, we zoom in on the Roissy-Orly area, and select all flights that overshoot, *i.e.* do not start or end at, this area (image (h)). When deleting these flights from the selection, and zoom out, we obtain the final desired result (image (i)).

3.1.2 Finding long flights

In our data model, each trajectory has a unique flight ID which is monotonically increasing over time. To find and analyze long-duration flights, we create a view with x mapping the day-time of each radar plot and y mapping the flight ID (Fig. 3 a). In this view, each flight is thus a horizontal line. The slope of the S-like shape formed by all lines indicates traffic dynamics along the day (amount of new flights appearing per time unit) We see here how traffic increases around 5 AM, stays constant during the day, and decreases around 10 PM. The width of this shape gives the average flight duration in our dataset (about 2.5 hours, the time required to cross France by airplane). In the same image, we see a few long horizontal lines. To understand these, we select them by brushing and plot them next in a top view (x=longitude, y=latitude, Fig. 3 b).

This view reveals particular eight-like shapes. Upon closer inspection, we discovered that these correspond to military planes that circle over the territory.

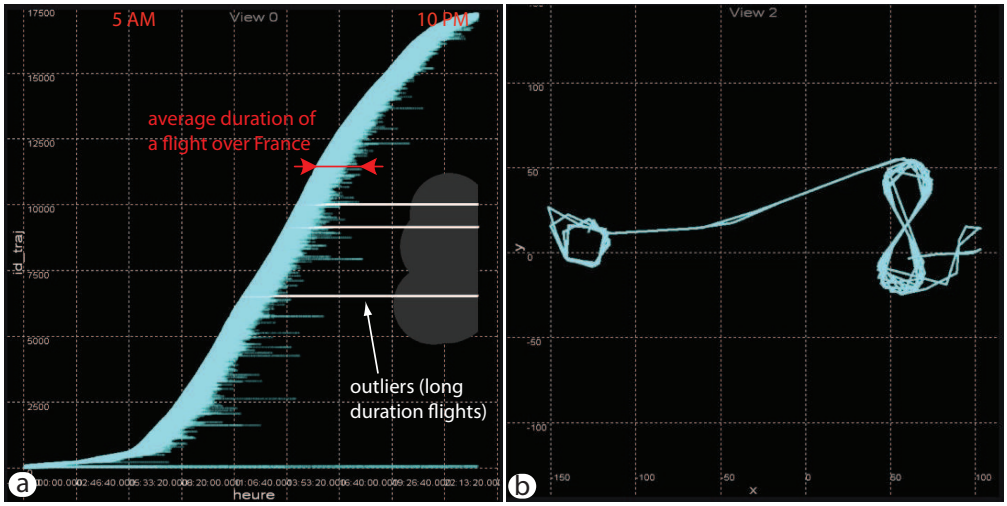

Fig. 3. Discovering long flights. (a) Flight duration *vs* flight ID. (b) Selection of long-duration outliers (military planes).

3.1.3 Advanced queries

A more complex data exploration scenario where FROMDADY was used by an ATC specialist is presented next. The user's task is to find instances of so-called *standard procedures*, *i.e.* aircraft that passes exactly over specific radar *beacons*, corresponding to referenced flight routes, is in a continuous climbing trajectory, and follows a well-defined flight direction.

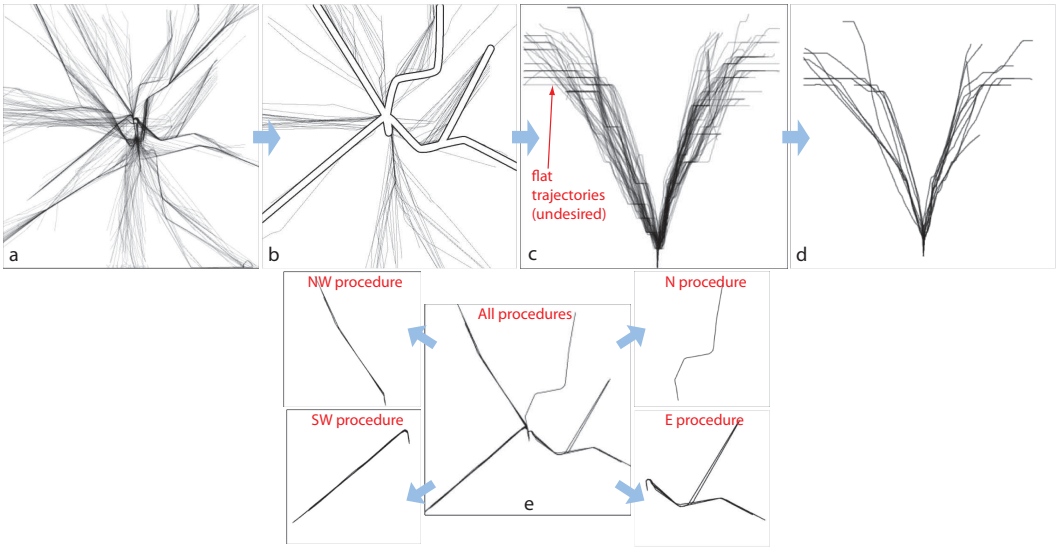

Fig. 4. Finding standard flight procedures. (a) All flights. (b) Flights *vs* flight routes. (c) Discovering ascending flights. (d) Selected ascending flights. (e) Standard procedure flights, further split into directions.

Aircraft do not always follow standard procedures. ATC controllers can shorten a trajectory for optimization reasons. Also, an aircraft can deviate from its trajectory if it overshoots beacons. The user has to filter out this kind of data, even though the criteria that define it are fuzzy.

To find standard-procedure flights, we proceed as follows (see Fig. 4). First, we plot all flights in a top view (image (a)). Next, we overlay the flight routes, as published by the air transportation authorities – for France, in this case (image (b)). As we can see, only a subset of flights falls within the flight routes. We next brush to select the outlier flights and delete them. Next, we display the remaining flights in a (x=latitude, y=altitude) view (image (c)). We now see several flights that contain a horizontal segment. As we are interested only in ascending flights, we select these and delete them from the view. This results in the flight subset shown in image (d)). We now show these flights again in a top view (image (e), center). We notice that, around the intersecting point (airport), these remaining flights are now well clustered along the flight routes, and follow several different directions. Finally, we iteratively brush over the flights' main directions, select all flights following each direction, and create four separate views (image (e), side views). Each such view contains our desired flights matching a separate standard procedure. The entire process took less than five minutes using the interactive queries of FROMDADY.

3.2 Wind parameters data mining

Wind is and important factor in aircraft traffic planning and control. ATC operators and analysts are interested to analyze the effects of weather, *e.g.* storms and strong wind currents. This helps in several use-cases, *e.g.* seeing how flight routes are disturbed by bad weather at a given time

instant; detecting airspace sector overload; seeing how weather at different moments in time (days) affects the same geographical location; and analyzing the effect of wind on different types of aircraft at different flying heights.

Of course, to answer all above questions, one would need to have an accurate data volume encoding wind speed and direction over the entire spatial and temporal domain of interest. This is challenging to obtain, given the very large space and time ranges that aircraft travel over *vs* the sparsity of weather stations and also *vs* the relative imprecision of weather simulations. For instance, the standard procedure in ATC today is to produce a so-called wind map every three hours for several altitude levels. However, this is not precise enough for accurate trajectory prediction with respect to weather factors.

A different use-case is to actually *measure* and predict wind data from aircraft flight measurements. The advantage here is that the latter can be obtained with high accuracy and over a dense sampling grid, *i.e.* at all space-time positions where aircraft is present. In other words, we consider aircraft as *sensors* that deliver wind information every few minutes. For example, Delahaye *et al.* present several methods where wind data is estimated from simulated aircraft positions (as if measured by radar) [11, 10, 9]. However, their method uses information from a *individual* aircraft, and thus suffers from undersampling. Moreover, they demonstrate their method only based on simulated aircraft position data.

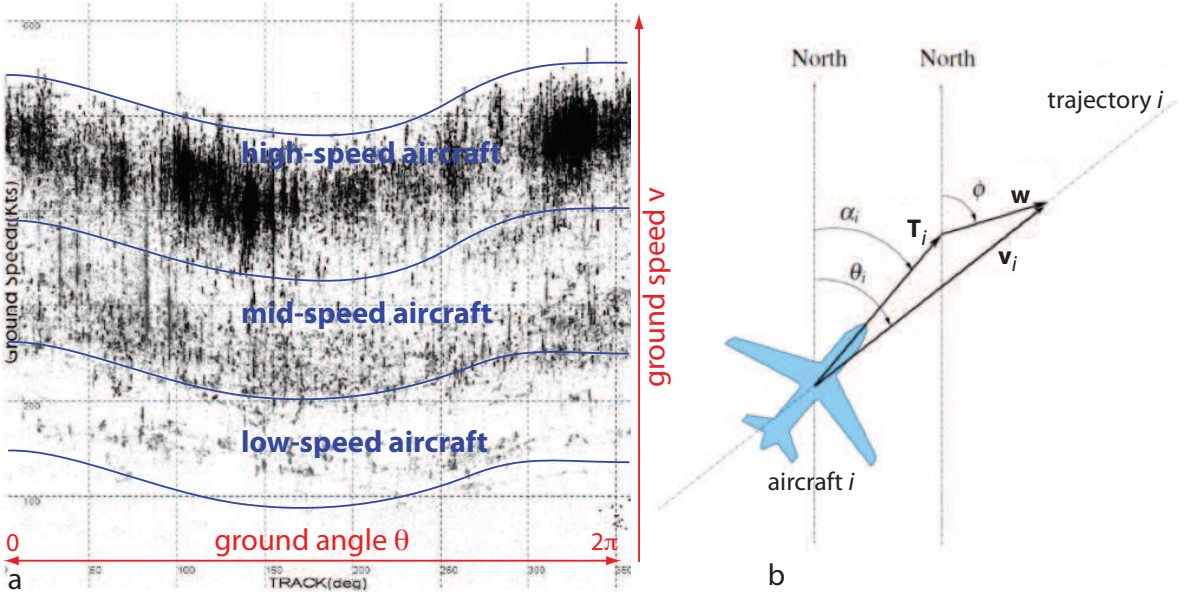

Fig. 5. Wind estimation from flight data. (a) Wind view scatterplot (for flight data in Fig. 2 b, Roissy region). (b) Wind model (see Sec. 3.2).

We show next how data mining and visualization tools can be combined to this end, by fusing information acquired from a dense set of aircraft trajectories, as measured by real-life ATC ground radar systems. Figure 5 (a) shows a so-called *wind view* produced with FROMDADY, for flights over the Roissy region, *July*5^{th} 2006. Denoting the velocity vector of an aircraft with respect to the ground (as measured by radar) by $\mathbf{v}$, the wind view maps the angle θ between $\mathbf{v}$ and the North direction to the x axis and the speed $v = |\mathbf{v}|$ to the y axis respectively. Drawing one point for each value (v, θ) for all aircraft flying over a given spatial region and time period, we thus obtain a scatter plot like the one in Fig. 5 (a). Points in this plot appear grouped in three sine-like dense bands. The densest (top) band contains high-speed planes, which are in the majority (regular commercial flights). The middle and bottom bands are slower aircraft, which are significantly less numerous (training and leisure flights, for example).

Let us analyze the meaning of these sine-like bands. Given a set of $i \in [1, N]$ flight trajectories, denote by $\mathbf{w}$ the wind vector (at an angle Φ with the North direction) and $\mathbf{T}_i$ the airplane's velocity, or cruising speed, with respect to the wind (at an angle α_i with the North), as in Fig. 5 b. Then we have that

$$\mathbf{v}_i = \mathbf{T}_i + \mathbf{w}$$

or, when projecting the above along the flight trajectory, and denoting $w = |\mathbf{w}|$ and $T = |\mathbf{T}|$,

$$v_i = T_i \cos(\alpha_i - \theta_i) + w \cos(\Phi - \theta_i). \tag{1}$$

Our aim is to compute w and Φ given several measurements of $\theta_i(t)$ and $v_i(t)$ for several moments t in a chosen time interval for several flights i. To proceed, we make the following observations. First, we consider only high-speed flights (top band in Fig. 5 a). These are, as we have seen, the most numerous data points. For such high-speed flights, the drift angle $\alpha_i - \theta_i$ is quite small: For an aircraft flying at *e.g.* $T_i = 450$ knots with a cross-wind of $w = 70$ knots, the error made when considering that $\cos(\alpha_i - \theta_i) \approx 1$ is about 1 percent of the aircraft speed. Secondly, we consider that for our time-interval of interest (*e.g.* tens of minutes), the wind is constant, *i.e.* w and Φ are not functions of t. Hence, we can simplify Eqn. 1 by replacing $T_i \cos(\alpha_i - \theta_i)$ by a unique speed $\bar{v}$ for all high-speed flights i, obtaining

$$v_i(t) = \bar{v} + w \cos(\Phi - \theta_i). \tag{2}$$

Next, by denoting $w_x = w \cos \Phi$ and $w_y = w \sin \Phi$ in Eqn 2, we obtain

$$v_i(t) = \bar{v} + w_x \cos \theta_i(t) + w_y \sin \theta_i(t) \tag{3}$$

Given a set of measurements $\theta_i(t)$ and $v_i(t)$ for a set of same-category (high-speed) airplanes flying over a given spatial region and time interval, we can now solve Eqn. 3 for $\bar{v}$, w_x, and w_y, using *e.g.* least squares methods. This gives us the sought wind speed magnitude w and direction Φ.

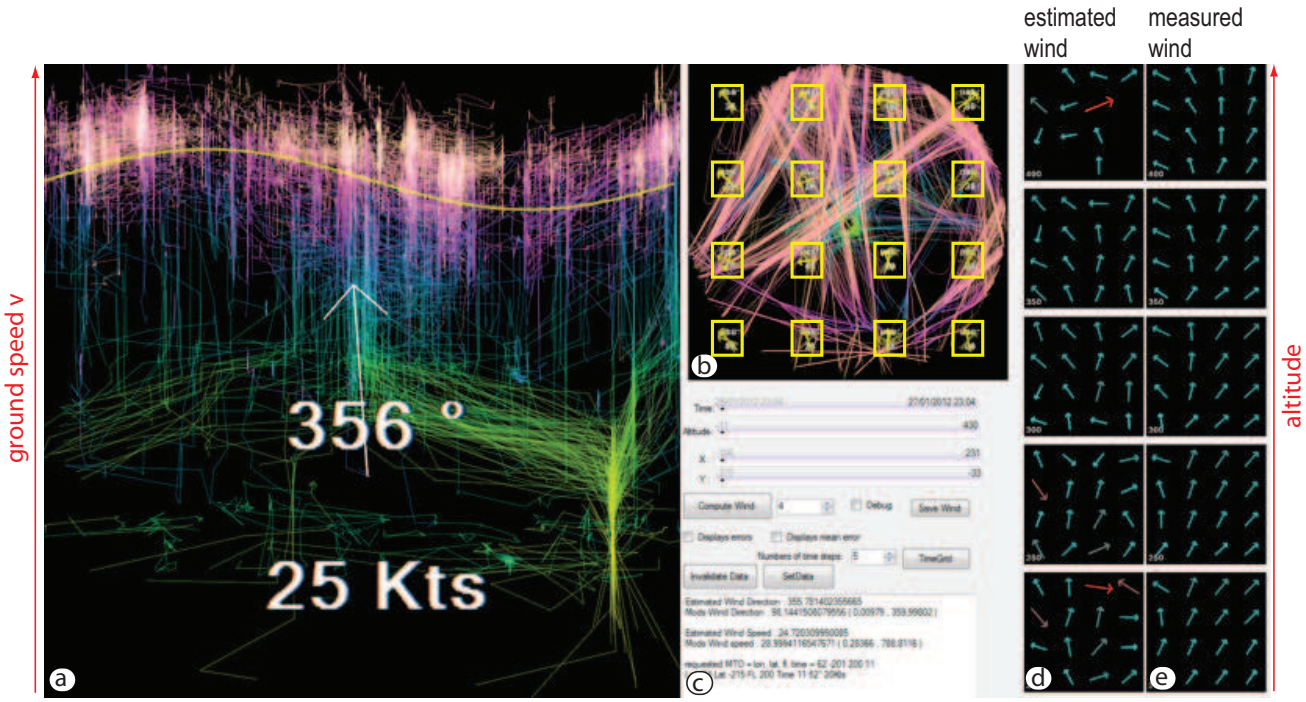

Fig. 6. Wind estimation visual analytics tool (see Sec. 3.2).

Let us note that this process is equivalent to fitting a single sine curve to all selected data points. The phase of this curve gives the wind direction Φ, while its amplitude is the wind speed w.

To apply the above automatic wind estimation, we however need to perform a robust selection of data points for aircraft speed $v_i(t)$ and directions $\theta_i(t)$ over a given space and time region. Obviously, solving Eqn. 3 depends on how many, and which, data points have been selected: If we 'mix' measurements for different classes of aircraft (*e.g.* high speed and medium speed), or select too few measurements, or select a too large spatial and/or temporal region, the resulting wind parameters will diverge more from the actual values.

We address the selection task by using an interactive visual approach, by extending FROMDADY's visual interface, as follows (see Fig. 6). First, we show a top view of the recorded trajectories, like in *e.g.* Fig. 2 b. This is our context view. Herein, users can select a spatial region of interest using brushing. Fig. 6 b shows such a selection corresponding to the southwest of France, which contains few climbing and/or descending trajectories. Such regions are the most robust for wind estimation, since they contain the most high-speed, cruising, flights (which is important given our earlier statement on relying on such data). Time intervals of interest can next be selected using a classical GUI based on range sliders (Fig. 6 c). Secondly, we display the data points in the selected time-space region of interest using the wind view. Additionally to the visualization in Fig. 5 (a), we color trajectories by their altitude using a yellow (low) to purple (high) colormap, and also connect consecutive points on a trajectory by lines (Fig. 6 a). Now, in contrast to the scatter plot in Fig. 5 (a), the three speed-bands are better visible. We also see that altitude correlates well with speed.

Given this wind view, users have now to select the data points to be fed to Eqn. 3 to compute wind parameters. For this, we provide two mechanisms. First, one can brush the wind view to explicitly select data points. For instance, in Fig. 5 (a), we would like to select the purple-and-blue points in the top high-speed band. Once these points are selected, we compute the wind parameters w and Φ as explained earlier, and display the results using text annotations and an oriented arrow glyph, see Fig. 5 (a). However, this solution strongly relies on the user's ability to precisely select the right data points from the wind view. An alternative solution we provide is to explicitly let the user fit a sine curve atop of the wind view. We allow users to fit such a curve (yellow in Fig. 5 (a)) by dragging it and changing its amplitude using the mouse, respectively the mouse wheel. Once the curve is fit, we can directly estimate the wind direction Φ as the curve's phase and the wind speed w as the curve's amplitude. The two interaction modes serve complementary purposes: Brushing is more fine-grained as it lets users explicitly select which data points should be considered in the wind computation, but requires more precision and time (on the average, two minutes to brush the desired points). Curve-fitting is coarser, as it relies on the user to visually fit the curve to the data, but is much easier to use in terms of interaction (on the average, seconds to fit a curve).

Given these mechanisms, we can easily compute and display wind parameters at many locations in our dataset. Figure 6 (b) shows several such locations, outlined in yellow, on the context top view. For each such location, we next compare our automatic estimations with 'ground truth' wind data produced by Meteo France, as follows. First, for each selected location, we subdivide the area around it into a $5 \times 5 \times 5$ regular grid (longitude, latitude, height). Next, we apply our wind estimation within each cell. Figure 6 (d) shows the results with arrow glyphs, colored by wind speed. Empty cells indicate locations where we have too few data points to be able to robustly estimate wind data, either automatically or by manual fitting. Figure 6 (e) shows the corresponding wind values measured by Meteo France. As visible, our estimation is not robust for high altitudes (too few data points) and low altitudes (plane speeds are too low and/or ascend or descend too much). However, for intermediate altitudes, our estimations match well the actual measured values.

To further evaluate our wind extraction process, we conducted informal discussions with air traffic controllers at ENAC [15]. Our goal was to assess the usefulness and validity of our wind extraction process and to understand how air traffic controllers use wind parameters in their aircraft monitoring tasks. The evaluation consisted of asking the controllers three questions: "How is the wind important in your daily work activity?"; "How do you retrieve wind parameters?"; and "How often do you verify wind parameters?". Next, we demonstrated and explained our visual tool, and asked questions about its effectiveness and efficiency.

Controllers validated the fact that wind parameters are important in their daily activity. However, when asked how they compute such parameters, the answers where not unanimous. They often retrieve wind parameters only when looking at aircraft behavior: 'When I compare how aircraft turn when facing north or south, I can assess the wind direction and the wind speed'. This estimation was recognized not to be accurate and only an approximation for dinging wind parameters. Controllers also have a screen which displays estimated wind parameters provided by Meteo France. This data is displayed in two tables with four geographic points each: south points (Barcelona, Montpellier, Nice, Ajaccio) and north points (Clermont Ferrand, Dijon, Lyon, Geneva), all at five flight levels. These tables are updated every six hours. However, as the controllers stressed, this information is simply too coarse-grained to be effective.

Controllers found our wind parameter computation method useful, technically sound, and more fine-grained than their current weather-station-based approach. They also noted that our tool is best not for ATC controllers who monitor aircraft, but rather for the so-called regulator controller who supervises traffic regulation and does not need to deal in real-time with aircraft. The regulator needs to forecast traffic evolution. Here, our dynamic wind parameter extraction can provide valuable information.

4 Density Map Techniques

When data amounts increase (*e.g.* by considering larger spatial regions or longer time periods), the visualizations presented in Sec. 3 become hindered by *clutter* – the presence of too many (overlapping) trajectories in a small spatio-temporal region. Clutter causes and reduction strategies in information visualization are discussed in [14]. Such strategies are similar to long-standing map generalization in cartography [5], concerned with legibly depicting a complex world in static, small-scale, 2D views.

Clutter is one of the main challenges to our visual scalability requirement (Sec.). In this section, we present several techniques for reducing clutter and thereby increasing visual scalability, based on *density maps*. The principle of density maps is simple: Consider a set of observations, or scattered data points, $\mathbf{x}_i$ contained in a spatial region $S \subset \mathbb{R}^2$, where each point has a data value $v_i \in \mathbb{R}^m$. We estimate a density function $\rho : S \to \mathbb{R}^m$

$$\rho(\mathbf{x}) = \sum_{\mathbf{x}_i} v_i \, K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) \tag{4}$$

where $K : \mathbb{R}^2 \to \mathbb{R}^+$ is a kernel function of bandwith h. Evaluating Eqn. 4 over S is known as kernel density estimation (KDE). Typical kernels used in practice are Epanechnikov (inverse quadratic) or Gaussian functions. The above process creates a continuous interpolation between the values v_i of the data points $\mathbf{x}_i$ over S. The resulting signal ρ is next displayed using classical techniques such as color mapping, height mapping, or contour lines. Evaluating ρ over a raster grid creates, thus, a dense-pixel visualization of the observations.

Density maps have several advantages for displaying point-based observations, as follows.

Visual scalability: By construction, a density map will emphasize strong concentrations of observations, and fade out sparse areas populated by a few spatial outliers. The kernel size h thus acts as a scale parameter: Observations much further apart than h will create small ρ values, whereas clusters of observations closer than h create high peaks in ρ. Setting h to a small fraction of the domain size S thus guarantees that large clusters are always visible, while sparse outliers are filtered out. Implicitly, this also reduces clutter.

Computational scalability: KDE can be easily and efficiently implemented using GPGPU image-based techniques. We experimented with two such implementations. In the first one, we store the kernel K as a scalar (luminance) OpenGL floating-point texture, and implement Eqn. 4 by drawing screen-aligned quads of size $2h$ centered at the points $\mathbf{x}_i$, using additive alpha blending. This method, also known as 'scattering' [25], is very simple to implement and can compute density maps for hundreds of thousands of data points over an image discretization of S of 1024^2 in roughly one second on modern graphics cards. In the second implementation, also known as 'gathering' [25], we used Nvidia's CUDA platform [41] to decompose Eqn. 4 in two 1D passes for separable kernels. This efficiently exploits CUDA's architecture and is about 10 times faster on the same platforms, thereby yielding interactive frame-rates for up to millions of data points.

In the following, we show several usages of density maps for the analysis of ATC datasets.

4.1 Detecting stationary trails

In this scenario, we aim to detect stationary trails, *i.e.* aircraft trajectories which 'loop' continuously over the same small spatial area over a day. Figure 7 (a) shows a typical 'top view' of the dataset in Fig. 1 (a), with a density map where ρ is mapped to color using a blue-to-red rainbow colormap. We now easily see dense regions as red spots on the map. These correspond to the main French airports, *e.g.* Roissy-Orly in this image. We also discover two circular shapes in the image. By construction (of the density map), these can only be areas where planes regularly looped over the same small spatial area.

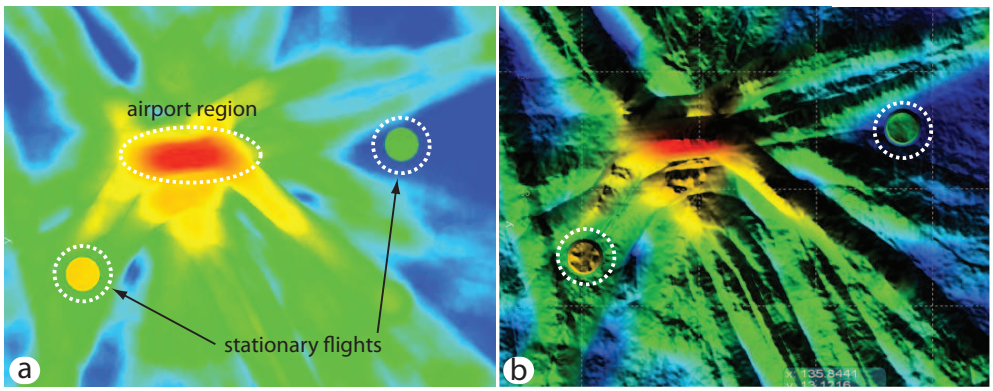

Fig. 7. Stationary trajectory estimation with color mapping (a) and color mapping and shading (b) (see Sec. 4.1).

However effective, color-mapped density maps exhibit little detail due to the inherent blurring caused by the filter kernel *and* to the fact that subtle color variations are not easily detectable by the eye. To address this, we now interpret the density ρ as height, and compute, at each image pixel, a shading or illumination value: Given a fixed light vector $\mathbf{l}$, shading is given by the dot-product $\mathbf{l} \cdot \nabla\rho$, following the well-known Phong illumination model. $\nabla\rho$ can be easily and efficiently computed by finite-differences over ρ. Figure 7 (b) shows the result of the shaded density map. Small-scale density variations are now much better visible. In particular, our two circular patterns emerge now better, since the density gradient acts much like an edge detector, and is not dependent on the choice of the color map used.

4.2 Data quality assessment

The quality of unknown datasets is a key issue for assessing the validity of an analysis. For instance, consider radar positions: ATC radars send data over networks with a constant stream rate (one radar position of each aircraft every 4 to 8 minutes). When using such positions to visualize aircraft trajectories (Sec. 3.1), or compute wind parameters (Sec. 3.2), we want to know if radar data is complete of if the data stream was possibly interrupted. We could use for this the visualization in Fig. 3 (a) which maps time to x and flight IDs to y. However, if data stream gaps are small (minutes), we would not see interruptions in the plotted horizontal lines, due to the limited screen resolution. Zooming in to regions where such gaps exist is a solution, but this implies we know where the gaps actually are, which is not true.

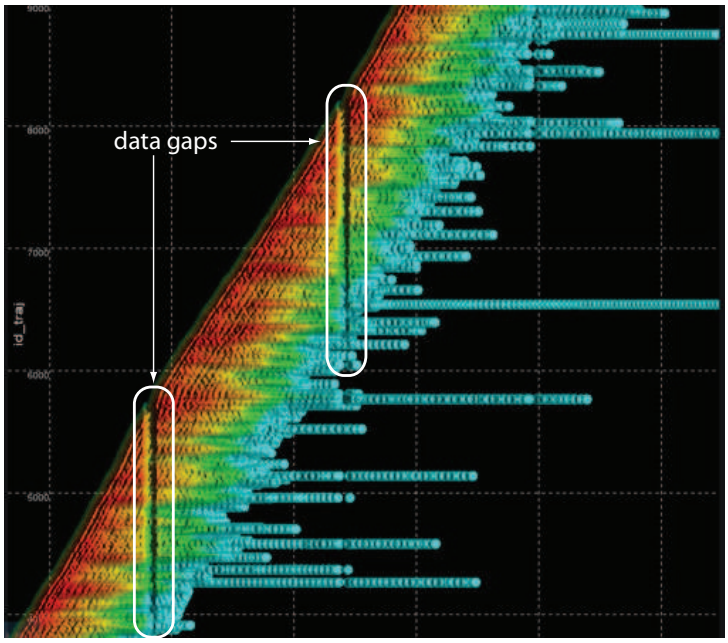

Fig. 8. Gaps in radar data found using shaded density maps (see Sec. 4.2).

Figure 8 shows the same view as in Fig. 3 (a), but now using the shaded density map of the recorded aircraft positions. To understand the results, observe that, for a continuous data stream having a *constant* sampling frequency (*e.g.* one point each four minutes), we obtain a constant spatial density of the plotted points. This will create flat plateau-like regions in the shaded density map. In contrast, areas where the sampling frequency drops will create sharp gaps (valleys) in the density map. Such valleys are further enhanced by the gradient-based shading. In Fig. 8, we see two such valleys. Since these are vertical, they indicate a failure of the radar to acquire position data for *all* considered aircraft at the *same* time moment. Secondly, since these valleys are thin, this indicates that the data glitch is short-lived (only a few minutes, in our case).

4.3 Abnormal traffic density detection

An important task for ATC controllers and regulators is to find abnormal traffic-density areas. These are spatio-temporal regions where many more flights occur than expected as based on flight procedures.

To find such regions, we use a mix of animation and linked views in FROMDADY, as follows (see also Fig. 9). First, we display an animated density map $\rho(t)$, computed as in Eqn. 4, but where the recorded aircraft points $\mathbf{x}_i$ are selected using a small time-window sliding over the extent of the considered dataset (one day in Feb. 2008). Secondly, we show the density map using a 3D height plot rather than a 2D density plot, since height is a more effective visual stimulus to attract attention to peaks (high density areas) than luminance [4]. Thirdly, we display a static 1D map showing how the average density (number of flights per total area of considered 2D flight domain) varies as function of time.

The average density map shows the expected flight-density trend over a typical day – few flights in the morning and late evening, and many flights overday. The five insets above show the instantaneous flight density $\rho(t)$ at five selected moments during the day. For each such view, we normalize $\rho(t)$ so that the user can easily distinguish low-density from high-density areas at the respective moment t. Correlating the average density map with the instantaneous density views lets us spot a salient outlier: Around 1 AM, we see a peak of the instantaneous flight density (Fig. 9, leftmost inset). Playing the animation, we see that this peak persists for roughly two hours. This indicates that many aircraft flew over around the same location around this time moment. In the same time, the average density map shows that there are very few flights at this time. Hence, the peak is clearly an outlier event. Further analyzing the flight IDs, we discovered that the peak was due to 12 aircraft who hung over the Belgium-France border for about two hours, not being able to land due to bad weather conditions. Some of these aircraft waited by performing elliptic cycle trails (called *stacking* in ATC vocabulary) at various altitudes for two hours until being able to land.

4.4 Flight safety analysis

An important part of the activity of air traffic controllers consists in maintaining safe distances between aircraft by giving clearance to pilots (heading, speed, or altitude orders). Air traffic is planned in advance: Companies request a flight plan from the regulatory authorities, which is

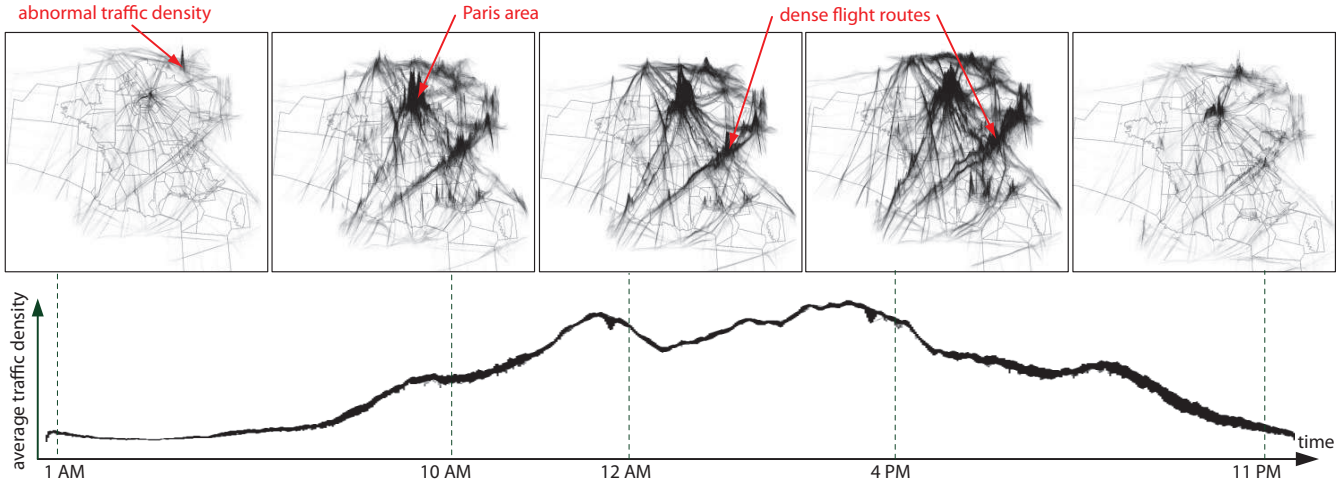

Fig. 9. Abnormal traffic density detection, French territory, Feb. 2008 (see Sec. 4.3).

next translated into a mandatory flight route. To do so, ATC controllers analyze the so-called safety net alarms on a yearly basis. These alarms trigger when aircraft fly below the safety distance. Such alarms are common since aircraft is flying in increasingly in dense areas, so planning cannot resolve all possible safety alarms in advance.

Visualizing the recorded alarms for a given time period can help ATC controllers discover unexpected and useful information. Figure 10 illustrates this. Here, we show a density map of the recorded alarms over the French territory in 2009. The density map is displayed using both color (blue=low, red=high) and transparency (transparent=low, opaque=high), so that high-density alarm areas stand out. We immediately see that the Paris region is densely packed with such alarms, which is not surprising, given the high expected traffic in this region. However, we discovered an unexpected dense-alarm zone around Montpellier, which has a much smaller airport, and thus should not have displayed such a high alarm rate. This finding triggered an official study of the flight routes of Montpellier, which led the regulatory agency to modify existing routes, and increase safety and capacity.

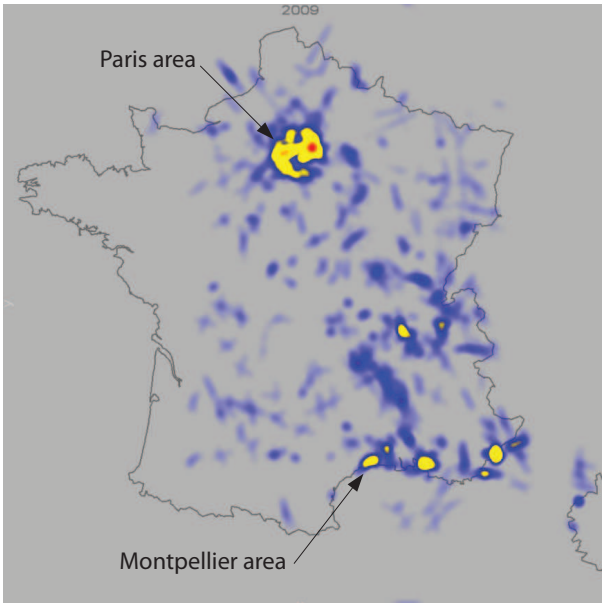

Fig. 10. Detecting flight safety problems (see Sec. 4.4).

5 Graph-based techniques

In Sec. 3, we showed how interactive queries can help in analyzing large sets of flight trajectories which are displayed in their 'raw' form. Next, in Sec. 4 we showed how visual scalability can be pushed further by aggregating spatially close data points using density maps. In this section, we take the quest for scalability via data aggregation a step further, and in a different direction. Rather than displaying the original flight paths, we simplify the view by using a powerful data aggregation technique: graph bundling.

Consider a graph $G = (V, E)$ with nodes V and edges E, and a graph layout L of G, *i.e.* a mapping from each node $\mathbf{n} \in G$ to a position $\mathbf{x}(\mathbf{n}) \in \mathbb{R}^2$. A typical drawing $D \subset \mathbb{R}^2$ of L consists of straight lines between the node positions $\mathbf{x}$. Figure 11 (e) shows such a drawing for a graph of airline connections in the US, where cities are nodes and edges are flight connections (235 nodes, 2099 edges). Clearly, such a drawing creates clutter up to a level that no insight can be obtained from it.

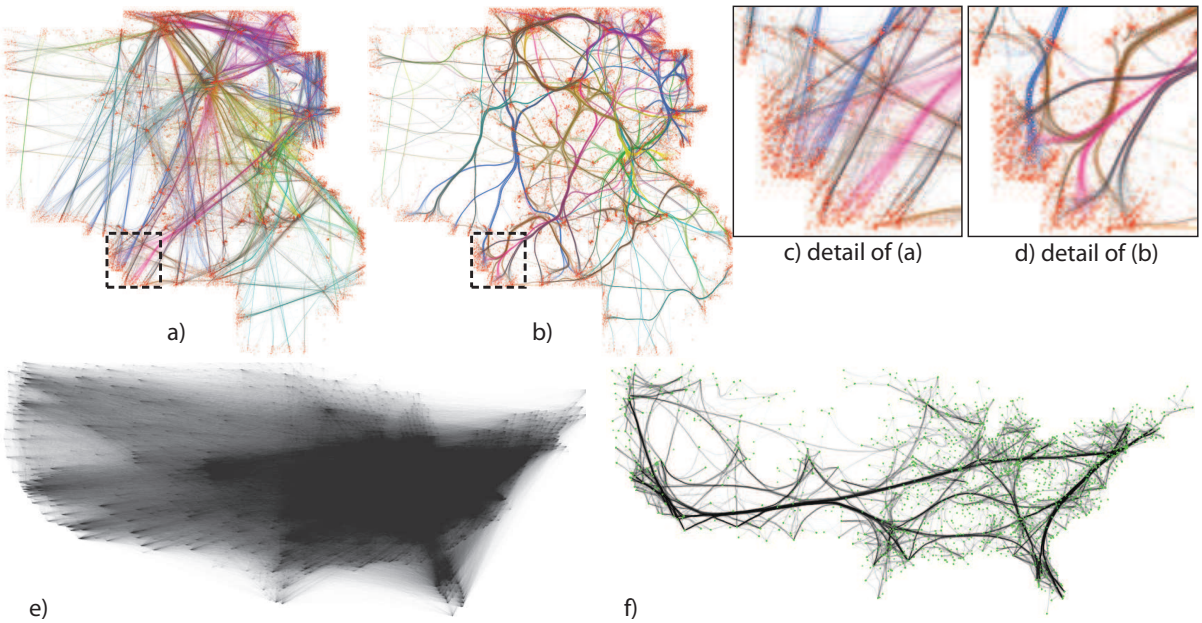

Fig. 11. Bundling for trajectory visualization. (a) Unbundled curved trajectory set. (b,d) Bundling of (a) using the SBEB method [16]. (e) Unbundled straight-line trajectory set. (f) Bundling of (e) using the KDEEB method [29]. See Sec. 5.

Graph bundling essentially lifts the core task addressed by an edge drawing of a graph to a higher level: In a classical graph drawing, users read, or visually follow, an edge to find out how *individual* nodes are connected to each other. In a bundled graph, users read the resulting drawing to find out how *groups* of close nodes are related to each other. To do this, bundling spatially gathers edges which connect such node groups into so-called *bundles*. Hence, bundling can be seen as an operator B that reads a given graph drawing D to produce a new drawing $B(D)$ in which edges connecting nodes who are close in D will also be rendered close.

Many bundling algorithms have been proposed in the last few years. Holten pioneered hierarchical edge bundling (HEB) for so-called compound graphs whose nodes are grouped in a hierarchy [26]. Dickerson *et al.* merge edges by reducing non-planar graphs to planar ones [12]. Gansner and Koren bundle edges in a circular node layout similar to [26] by area optimization metrics [22]. Dwyer *et al.* use curved edges in force-directed layouts to minimize crossings, which implicitly creates bundle [13]. Force-directed edge bundling (FDEB) creates bundles by attracting edge control points [27], and was adapted to separate opposite-direction bundles [46]. MINGLE uses multilevel clustering to accelerate the bundling process [21]. Flow maps produce a binary clustering of nodes in a directed flow graph to route curved edges [43]. Control meshes are used to route curved edges [44, 58], a Delaunay-based extension called geometric-based edge bundling (GBEB) [8], and 'winding roads' (WR) which use Voronoi diagrams for 2D and 3D layouts [35, 34]. Skeleton-based edge bundling (SBEB) uses the skeleton of the graph drawing's thresholded distance transform as bundling cues to create strongly ramified bundles [16]. Kernel density bundling (KDEEB) computes an edge density map and iteratively shifts edges towards their local density maximum to achieve bundling [29].

However, in nearly all these works, bundling was applied to straight-line graph drawings. For our ATC domain, we actually have more information: Considering the take-off and landing positions of aircraft as nodes, we can build a graph whose edges are the actual flight routes. Note that, strictly speaking, such a dataset is a graph where each node is connected to a *single* other node. Moreover, edges are not straight lines, but actual curves in $\mathbb{R}^3$. The question is: How can graph bundling help us in understanding such datasets?

Figure 12 shows the application of several recent graph bundling algorithms to several datasets consisting of airline and transportation trajectories. As visible in all images, bundling effectively creates white-space gaps between groups of densely-packed trajectories, while in the same time grouping such trajectories together. The resulting bundled images are effective in showing how *groups* of closely-spaced airports are connected to each other. In other words, bundling exposes the coarse-grained structure of the connection network induced by the considered flights. However, bundling will necessarily make it hard to spot outlier trajectories from the visualizations. To highlight such trajectories, other techniques would be needed.

A second insight from this image is that different bundling algorithms produce slightly differently-looking bundling results on the same dataset – compare *e.g.* images (a,b) or images (c-h) in Fig. 12. Although the 'backbone' connections are visible, and look similarly, in the considered images, small variations in terms of curvature, position, and twisting of the bundles are visible. However, we argue that this is not a main issue for the core use-case of bundling. Indeed, the *key* purpose of bundling graphs is to help users assess easier how groups of spatially-close nodes are connected to each other. as such, the precise spatial position of a bundle is not important – its topology is.

An interesting connection exists between graph bundling and density maps. Fundamentally, bundling can be seen as an image-space operator which enhances the edge-density signal as it moves close edges even closer, and thus makes inter-edge gaps even larger. This observation has been directly exploited by [29], which implement bundling directly by sharpening the edge density computed using kernel density estimation (KDE, Eqn. 4). Since KDE can be very efficiently implemented using GPGPU techniques (Sec. 4), graph bundling naturally inherits this property. For instance, bundling the French airlines graph (Fig. 12 b), which has 34.5K nodes and 17.2K edges, takes around 0.1 seconds on an NVidia 330M graphics card, and 20 milliseconds on an Nvidia 690 GTX card. Interestingly, we observe that this process is nothing but the well-known mean-shift operator presented more than a decade ago in a completely different application domain: image segmentation [7]. We see here yet another instance of our earlier observation that image-based techniques are a strong instrument in producing visually scalable visualizations of Infovis data.

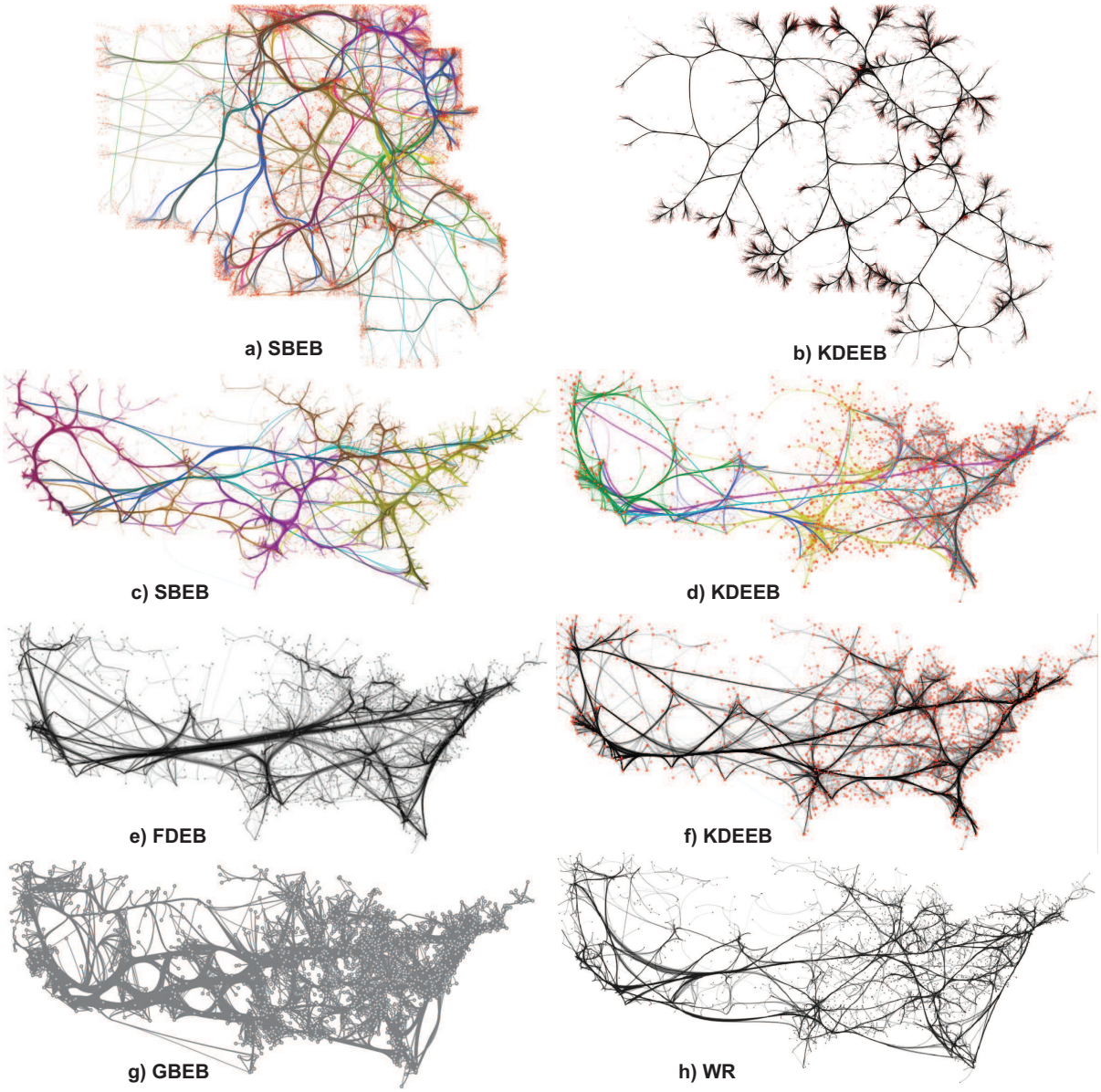

Fig. 12. Comparison of different graph bundling algorithms for trajectory visualization. (a,b) French airline trajectories over one week. (c-h) US airlines trajectories (see Sec. 5).

6 Animation Techniques

So far, all our examples (with the exception of the animated density map in Sec. 4.3) have used static visualizations. Such visualizations have the advantage that they are relatively easy to follow, as they can be examined at the desired pace as set by the user. However, dynamic, or animated, visualization techniques have also important advantages. First, we can use animation to increase visual scalability in cases where static visualization techniques do not offer enough space to display all available data, or do so with too high clutter. Secondly, animation is the natural technique to convey data that changes dynamically.

In this section, we present several uses of animation to support the visual analysis of air-traffic datasets: focus-and-context exploration (Sec. 6.1), dynamic bundling (Sec. 6.2), and flow visualization (Sec. 6.3).

6.1 Focus-and-context exploration

Consider a bundled visualization of a large set of airline trajectories, using one of the graph bundling techniques presented in Sec. 5. Figure 13 (a) shows such a visualization, for the French airline trajectories over a whole week. Although, as mentioned earlier, bundling helps in finding the connectivity pattern between airports induced by flights, there are also drawbacks: Several spatially different trajectories are now grouped by bundling. This creates two problems, as follows.

Spatial distortion: We cannot see any longer the *geographical* information originally present in the trajectories, since bundling has distorted this information.

Overdraw: Bundling inherently decreases clutter but increases overdraw – in other words, we can now see a bundle that contains many trajectories, but have no way to examine these trajectories separately, *e.g.* examine their flying heights or flight IDs.

We emphasize these problems in Fig. 13 (a) by coloring the bundled edges by local edge density (estimated using Eqn. 4), using a blue-to-red colormap. As visible, several bundle segments appear red. In these areas, the overdraw problem is most severe, so mechanisms are needed to explore the many bundled edges.

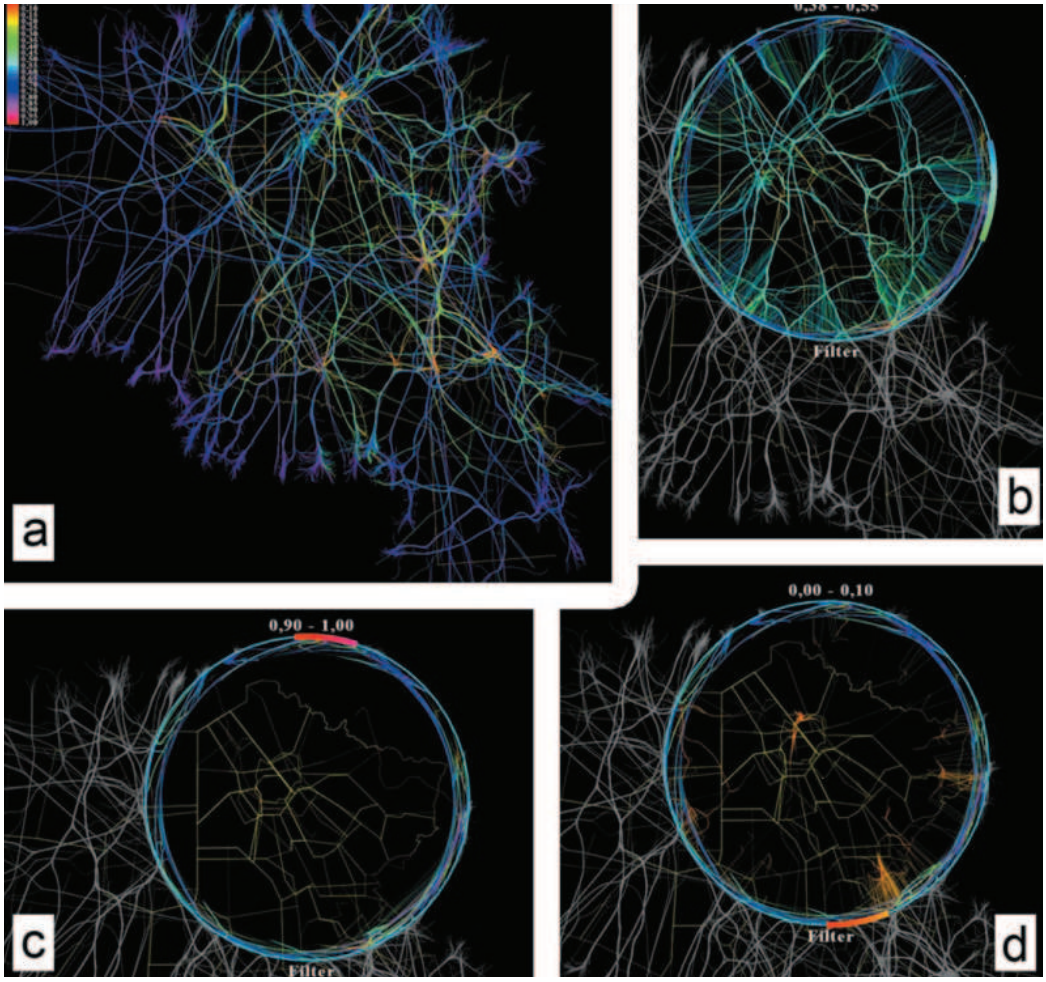

Fig. 13. Focus-and-context exploration of bundled airline trajectories (see Sec. 6.1).

We cannot (easily) adapt bundling as a method to fully cope with the two above-mentioned problems, at least not if we want to keep its key advantage of de-cluttering dense trajectory datasets. However, we can exploit animation to solve these problems *locally*. Such solutions are globally known under the names of *semantic lenses* and *focus-and-context* (F+C) techniques. The key idea is to locally deform, or distort, a given visualization so that, on the one hand, hidden information becomes better visible, but the overall structure implied by the original visualization is preserved. For example, the dust and magnet technique allows users to de-clutter large scattered plots by placing several data-attribute-driven 'magnets' in the display space and moving data points close to them based on the points' attributes [57]. In the earlier-mentioned vessel visualization of Scheepens *et al.* [45], a simple form of semantic lens is used to emphasize specific trajectories, *e.g.* slow moving ships, by tuning the shading and blending parameters. However, spatial deformation is not used to declutter trajectories, since position data is deemed too important to be altered. Many variations have been proposed from the original fisheye view [19]. For graphs, techniques include local edge deformations, or re-layouts, such as the EdgeLens and its variations [56], and selective edge hiding based on attributes at the position of a user-specified focus point. The local edge lens and bring-neighbors lens [52] are variations of EdgeLens which remove edges between nodes within a focus zone (lens) and pull nodes connected to nodes-in-focus within the lens, respectively. Edge plucking allows the user to explicitly drag groups of edges away to clarify cluttered zones and/or specify nodes or edges to be left unmoved [54, 55].

For our ATC datasets, we explored several types of semantic lenses and F+C deformation. The optimal result was given by using animation, as follows (for details, see [28]). Given a bundled trajectory visualization, we first select a region of interest, or focus, by moving the mouse to that area on the screen. The size of the region of interest is controlled by using the mouse wheel, and visually indicated by a circle centered at the focus. Next, we select a data-range of interest. For example, we want to focus on trajectories flying within a certain height, or speed, or direction range. We select these using again the mouse wheel with the shift key pressed. When next clicking the mouse button, all trajectories that fall within the focus region but are *not* within the data-range of interest, are smoothly pushed away from the focus region. This leaves only the trajectories of interest inside the focus area, and thus addresses the overdraw problem mentioned earlier.

Figures 13 (b-d) show three images from such an exploration, for different altitude ranges and for the same focus region. As visible, the focus region becomes much less cluttered as compared to the initial image (a), since all edges not falling within the altitude range of interest have been pushed out of the lens. This allows us to precisely answer the question whether flights with certain heights are present in a given

spatial region – or, put simpler, it allows us to 'dig' within a bundle to find out its contents. Identical procedures can be used to *e.g.* find out flights with specific flight IDs or speeds in a given spatial region.

However, this method still keeps trajectories being deformed. Indeed, trajectories outside the lens or inside the lens and the data-range of interest stay bundled (thus, deformed away from their original geographical positions). Also, trajectories which are pushed away from the lens are deformed by the F+C technique. To reveal the original geographical information, we use the same F+C lens idea, but in a different way. When activating the lens (by clicking the mouse), we now smoothly deform the trajectories inside the lens between their bundled positions and their unbundled (geographically correct) positions. Figure 14 illustrates this. In the left image, we activate the lens over the central area of our French airlines dataset. The insets below the figure show three frames from the animation between the bundled and unbundled trajectories. To outline the focus area more, we also draw all trajectory segments outside the lens in gray and keep the ones inside the lens colored (in this case, by altitude). This helps the user in locally seeing the geographically correct trajectory data, but still having an uncluttered context view (given by the bundling). The right image shows the inverse process: Here, we display the unbundled, geographically correct, set of trails as the context. Within the lens, we smoothly animate the trails towards the bundled view. This helps the user having the geographically correct trajectory context, but locally reducing clutter by bundling. Summarizing, this second lens mode addresses the spatial distortion problem mentioned earlier.

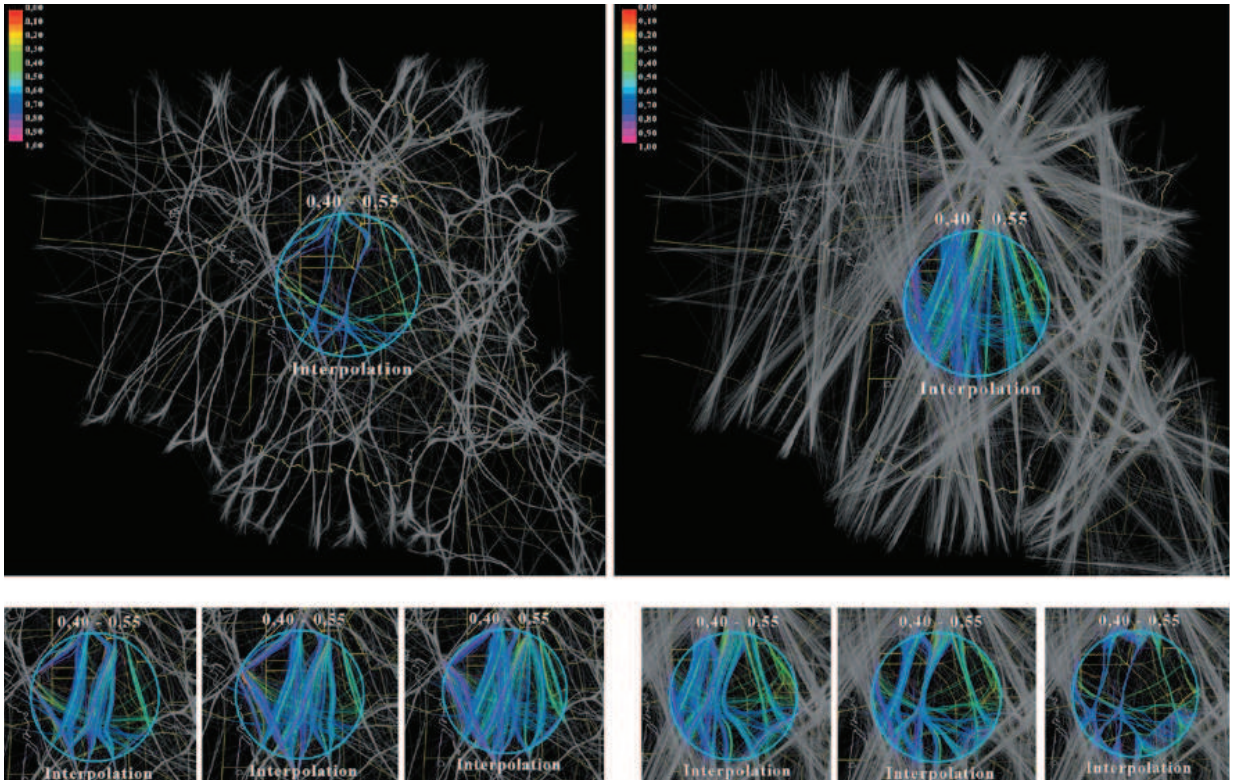

Fig. 14. Smooth navigation between bundled and unbundled trajectories. Left: local unbundling. Right: local bundling (see Sec. 6.1).

Image-based techniques are a crucial aspect in the implementation of our F+C-and-semantic lens. For its first mode (overdraw task), the lens requires as input only a set of spatial trajectories (either bundled or not). The smooth animation that pushes out-of-data-range trajectories outside of the lens is implemented by computing a radial distance field from the lens boundary inwards [28], and incrementally moving trajectory points in the opposite direction of this field's gradient. Computing this distance field can be done in real-time even for complex lens shapes, using CUDA-based techniques [48]. The second lens mode (spatial distortion task) requires having the bundled and unbundled versions of a trajectory set. Computing the bundling is real-time, as explained in Sec. 5. Computing the animation between the two trajectory sets is also real-time, as it requires only a simple linear interpolation between corresponding trajectory points.

6.2 Airline trajectory dynamics

The examples presented so far illustrated how we can analyze a *given* set of flight trajectories. Such scenarios are effective when trajectories are known in advance, *e.g.* when assessing data recorded in the past. However, in the ATC context one also wants to analyze data that comes 'live' within a monitoring system – an instance which we further call *streaming* data.

All interactive exploration techniques of FROMDADY presented in Secs. 3 and 4 clearly work as well for streaming data, by using a sliding window technique that continuously feeds the visualization system with measurements around the current time moment. However, graph bundling techniques (Sec. 5) are more challenging here. Indeed, many of these techniques are required to have an entire graph in advance to perform bundling. Moreover, there is no guarantee that bundling graphs which are relatively similar, *e.g.* obtained by considering trajectories within two consecutive instances of a sliding time window, will yield relatively similar bundles. In other words, the considered bundling operator B is not Cauchy-continuous. If this is not the case, bundling may exhibit discontinuities which are both disruptive and misleading (they show change where there is none).

Studying the existing bundling algorithms mentioned in Sec. 5 reveals that this is indeed a valid concern. HEB cannot be used for dynamic trajectories, since it requires having a meaningful end-point hierarchy. WR, MINGLE, and GBEB rely upon various clustering operations or spatial partitioning (Voronoi diagrams) which are not Cauchy-continuous. SBEB requires both pre-clustering of similar trajectories, and uses

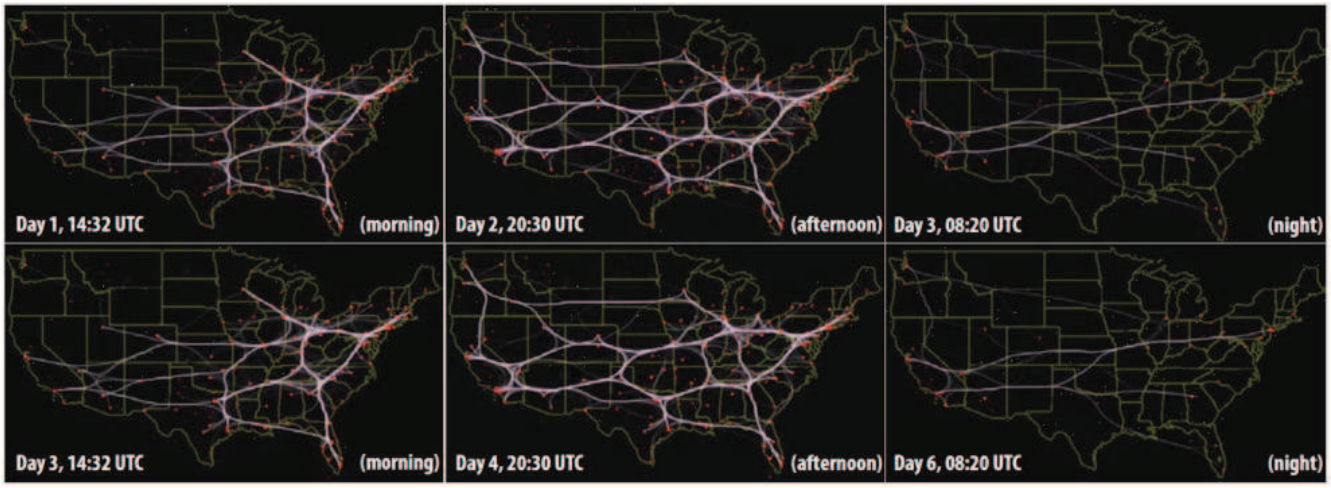

Fig. 15. Dynamic bundling of streaming trajectory datasets, US airline dataset, one week (see Sec. 6.2).

medial axes, which are quite sensitive to small variations in the input geometry. However, KDEEB provides here an ideal solution: As it is based on the mean-shift procedure [7], which is continuous in the density map input, KDEEB will also be continuous if this density map changes continuously in time. This is easy to achieve by updating the trajectories' density map continuously as new trajectories enter and/or exit the sliding time window [30]. Note that, from an image-processing perspective, this is equivalent to applying a low-pass filter on the instantaneous density map $\rho(t)$ with filter width given by the time window size.

Figure 15 shows the results of applying the time-dependent KDEEB bundling on a streaming trajectory dataset containing a full week of flights over the US. As visible in the stills (and the submitted video), the emerging bundles evolve continuously. More importantly, the animation allows us to recognize the 'core' connectivity pattern between US cities which stays stable during the week, and separate it from flights which change significantly over different days. In Fig. 15, for instance, we see that the morning, midday, and night flight patterns are quite similar for different days of the considered week. However, patterns for different day moments are quite different: In the morning, we see how flights concentrate mainly on the US East Coast. At midday, the flight pattern uniformly covers the entire US territory. During the night, we see only a few long-haul coast-to-coast flights being present.

6.3 Flow visualization

The visual analyses presented so far focused on intermediate and coarse levels of the dataset, *e.g.* displaying entire trails or even aggregating several such trails over space or time into even simpler metaphors such as density maps (Sec. 4) or bundles (Sec. 5). However, apart from such exploration levels, ATC datasets are fundamentally built from low-level information on the *instantaneous* positions of aircraft. Visualizing such fine-grained information can give local insight into the dynamics of flight patterns.

Classical ATC systems achieve this by showing actual aircraft locations in real-time on their screens, using glyphs to map position, speed, heading, height, and aircraft ID. While this works well for small-scale spatial regions, displaying glyphs at larger geographical and/or time scales quickly produces too much clutter. To achieve the fine-grained insight, and still maintain visual scalability, we revert once more to image-based techniques in an animation setting, as follows (see also Fig. 16).

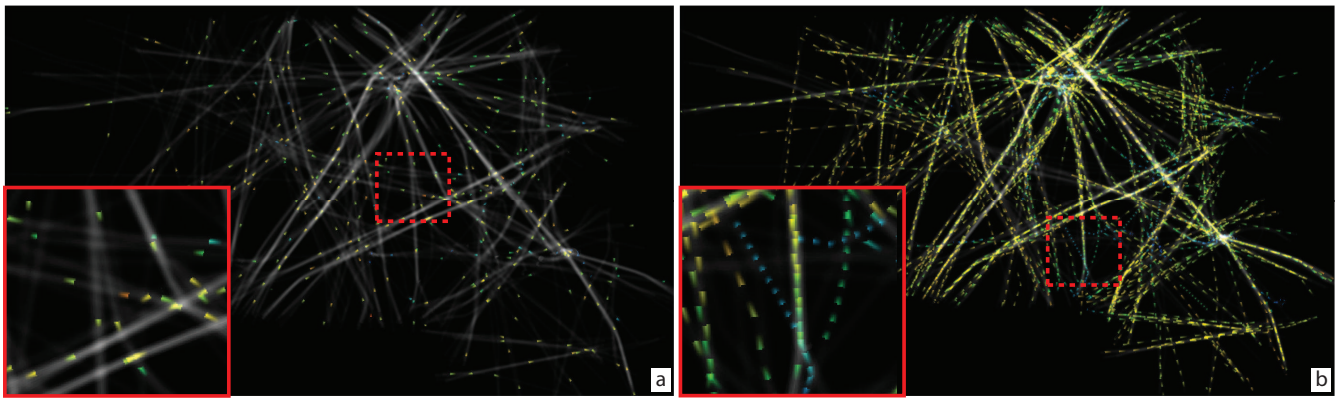

Fig. 16. Aircraft flow visualization using animated textures, one-week French airlines dataset (see Sec. 6.3).

As background for our visualization, we display the entire trajectories in the considered dataset in gray, using alpha blending. However, in contrast to *e.g.* Fig. 2 (b) which uses the same principle, we now blur the trails with a small (3×3) Gaussian filter. This helps pushing the trajectories away from the focus visualization (to be discussed next). Next, we consider a sliding time-window $W(t) = [t - \Delta t, t + \Delta t]$ centered at the current moment t, and select all data points from out input dataset which fall within $W(t)$. This yields several trajectory segments. We next construct, for each segment, a Gaussian filter profile $\Psi(t)$ centered at t and which decays to a very small value over the time-window width 2Δ.

Finally, we construct a transparency texture over the trajectory segments by multiplying Ψ with a saw-like profile given by a one-sided Gaussian filter ψ whose width is identical to the distance between two consecutive positions of a plane on its trajectory. The emerging effect consists of long pulses (of length 2Δ) that travel along the trajectories in the direction of the underlying flights, and which are formed by trains of short arrow-like pulses (of length δ) – see insets in Fig. 16). The short arrow-like pulses serve to indicate the *direction* of the aircraft motion. The long pulses ensure spatio-temporal continuity, *i.e.* help us follow the aircraft as they move along their trajectories.

Tuning several visualization parameters can help emphasizing different aspects. For instance, in Fig. 16 (a), we set $\Delta = \delta$, *i.e.* only draw the instantaneous plane positions as oriented arrows. Color indicates here flight altitude on a blue-to-red colormap. This view is best when users are interested to follow individual planes within a small spatial region. In Fig. 16 (b), we set Δ to about two hours. The lengths of the short pulses effectively indicate the planes' speeds (longer pulses show faster moving planes). For instance, in the inset we see a fine-grained blue trail segment indicating a slow, low height, outlier flight in an area with fast (long pulse) and higher (green) flights.

The proposed technique bears some conceptual similarities with another well-known image based technique for visualizing flow fields: Image based flow visualization (IBFV) [53]. Given a 2D vector field, IBFV constructs textured pulses that are locally tangent to, and which move along with, the flow. Animation is the key element in conveying the flow impression to the user. Our technique also conveys the aircraft motion by means of animating a texture. However, many differences exist. While IBFV generates a random-noise 2D texture, we use a regular (train-pulse) 1D texture. More importantly, IBFV assumes that at each point in the considered spatial domain, there is a single direction of the input vector field to visualize. In our case, this is not possible, since there may be several planes flying in different directions at the same 2D screen point (*e.g.* planes crossing each other at different heights).

7 Discussion

From our presentation of the use of image-based techniques for ATC flight data visualization, several general points can be made, as follows.

Computational scalability: Image-based techniques are highly scalable, and allow generating visualizations that capture the motion of thousands of planes over large spatio-temporal extents in real-time using modern GPGPU techniques. As the computing performance of graphics cards keeps increasing, we foresee that our approaches will remain highly scalable from this perspective even for larger data volumes. Since modern graphic cards can run proprietary code [42] to perform computational tasks on a GPU, *i.e.*, perform general purpose computation on the GPU (a process called GPGPU) [51], they are an effective and easy-to-use source of massively parallel computation power.

Visual scalability: In contrast to classical ATC visualizations which use geometric primitives such as glyphs and annotations, image-based and pixel-based techniques scale much better to display flight data over large spatial domains. The usage of spatial aggregation by means of density maps, filtering, and bundling only increases this scalability, and also implicitly reduces small-scale clutter. However, the price to pay here is the inability to display large amounts of information for a single flight. As such, image-based approaches seem best suited for creating *overviews*, while classical glyph-based approaches are best for displaying detail information. We foresee thus that the two types of techniques will coexist, as they serve complementary purposes.

Multivariate data: Current image-based techniques are limited in being able to show only a few independent variables per pixel, *e.g.* encoded in luminance, hue, lighting, and texture. However, ATC datasets have more information, in the range of ten or even more such variables per measured point. Extending image-based techniques to display more variables is an open challenge.

2D *vs* 3D: All techniques presented here, with the exception of the height plots for density (Sec. 4.3), are essentially 2D techniques. An open question is whether 3D image-based techniques can bring extra added value *e.g.* in terms of being able to display more data or data variables, but also for reducing ambiguities created by overdraw. For instance, the bundling techniques presented here (Sec. 5) could be extended to incorporate height information. Although this is relatively straightforward to do from a technical perspective, the open question is whether such 3D visualizations are as effective as their 2D counterparts.

Usability: We have tested all our visualization by involving ATC controllers [15] in both their design process and actual evaluations using typical ATC use-cases. Initial results have been very promising. However, a formal user evaluation still lacks, and is needed to better quantify the advantages and limitations of our techniques, and also to assess them *vs* the larger set of ATC tasks mentioned in Sec. .

Generalization: Although our current applications have focused only on ATC datasets, the presented techniques and approaches are not designed with constraints specific to ATC data. As such, they could be easily applied to other use-cases involving different types of transportation-related datasets, *e.g.* the visual analysis of vessel or pedestrian motion.

8 Conclusion

In this paper, we have presented the usage of image-based information visualization techniques for the analysis of datasets emerging from the air traffic control (ATC) domain. Image-based techniques propose different data depiction metaphors than traditional glyph-based methods. As each screen pixel is individually synthesized to reflect data values, image-based techniques are more visually scalable than their glyph-based counterparts, offer an easy way to construct aggregated (simplified) visualizations, and map efficiently and naturally to GPGPU-accelerated implementations. We have illustrated the application of image-based techniques in the ATC domain by means of four types of such techniques: interactive exploration, density maps, graphs, and animation. For each technique type, we presented several use cases, supported with real-world ATC data.

Several future work directions exist. From a technical perspective, the current palette of image-based techniques can be further extended to display more variables per point, thereby increasing scalability. Secondly, the existing techniques can be refined and composed to support several new scenarios and use-cases in the ATC application domain. Finally, from a theoretical perspective, new connections can be established between well-established visualization techniques for vector and tensor fields and use-cases in the ATC domain, thereby enlarging the set of visualization options for air traffic specialists with more powerful and more versatile data exploration options.

References

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

[2] G. Andrienko, N. Andrienko, and M. Heurich. An event-based conceptual model for context-aware movement analysis. *Int. J. Geogr. Inf. Sci.*, 25(9):1347–1370, Sept. 2011.

[3] K. Beard, H. Deese, and N. Pettigrew. A framework for visualization and exploration of events. *Information Visualization*, 7(2):133–151, 2008.
[4] J. Bertin. *Sémiologie Graphique. Les diagrammes, les réseaux, les cartes*. Gauthier-Villars, 1967.
[5] B. Buttenfield and R. McMaster. *Map Generalization: Making rules for knowledge representation*. J. Wiley & Sons, 1991.
[6] F. Chevalier, P. Dragicevic, and C. Hurter. Histomages: Fully synchronized views for image editing. In *Proc. ACM UIST*, pages 281–286, 2012.
[7] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603–619, 2002.
[8] 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.
[9] D. Delahaye and S. Puechmorel. Aircraft local wind estimation from radar tracker data. In *Proc. ICARCV*, pages 1033–1038, 2008.
[10] D. Delahaye and S. Puechmorel. TAS and wind estimation from radar data. In *Proc. Digital Avionics Systems Conference (DASC)*, pages 2.B.5–1–2.B.5–16. IEEE, 2009.
[11] D. Delahaye, C. Rabut, and S. Puechmorel. Wind field evaluation by using radar data and vector spline interpolation. In *Proc. ICCA*, pages 219–224, 2011.
[12] M. Dickerson, D. Eppstein, M. Goodrich, and J. Meng. Confluent drawings: Visualizing non-planar diagrams in a planar way. In *Proc. Graph Drawing*, pages 1–12, 2003.
[13] T. Dwyer, K. Marriott, and M. Wybrow. Integrating edge routing into force-directed layout. In *Proc. Graph Drawing*, pages 8–19, 2007.
[14] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE TVCG*, 13(6):1216–1223, 2007.
[15] ENAC. National school of civil aviation, 2013.
[16] 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.
[17] Eurocontrol. NEST: Network Strategic Tool, 2013. www.eurocontrol.int/articles/airspace-modelling.
[18] J. D. Fekete and C. Plaisant. Interactive information visualization of a million items. In *Proc. Infovis*, pages 117–125, 2002.
[19] G. Furnas. Generalized fisheye views. In *Proc. ACM CHI*, pages 16–23, 1986.
[20] GAIN Group. Guide to methods and tools for airline flight safety analysis, 2004. flightsafety.org/files/analytical_methods_and_tools.pdf, 2^{nd} ed.
[21] E. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. In *Proc. PacificVis*, pages 187–194, 2011.
[22] E. Gansner and Y. Koren. Improved circular layouts. In *Proc. Graph Drawing*, pages 386–398, 2006.
[23] H. Gaspard-Boulinc, Y. Jestin, and L. Fleury. Epoques: a user-centerd approach to design tools and methods for atm safety occurences treatment. In *Proc. ESReDA (European Safety, Reliability and Data Association)*. European Commission, 2003.
[24] R. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.
[25] B. He, N. Govindaraju, Q. Luo, and B. Smith. Efficient gather and scatter operations on graphics processors. In *Proc. ACM/IEEE Supercomputing*, pages 238–245, 2007.
[26] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE TVCG*, 12(5):741–748, 2006.
[27] D. Holten and J. J. van Wijk. Force-directed edge bundling for graph visualization. *CGF*, 28(3):670–677, 2009.
[28] 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.
[29] C. Hurter, O. Ersoy, and A. Telea. Graph bundling by kernel density estimation. *CGF*, 31(3):435–443, 2012.
[30] C. Hurter, O. Ersoy, and A. Telea. Smooth bundling of large streaming and sequence graphs. In *Proc. PacificVis*, pages 374–382. IEEE, 2013.
[31] C. Hurter, R. Lesbordes, C. Letondal, J.-L. Vinot, and S. Conversy. Strip'tic: Exploring augmented paper strips for air traffic controllers. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 225–232, New York, NY, USA, 2012. ACM.
[32] 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.
[33] D. A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE TVCG*, 6(1):59–78, 2000.
[34] A. Lambert, R. Bourqui, and D. Auber. 3D edge bundling for geographical data visualization. In *Proc. Information Visualisation*, pages 329–335, 2010.
[35] A. Lambert, R. Bourqui, and D. Auber. Winding roads: Routing edges into bundles. *CGF*, 29(3):432–439, 2010.
[36] P. Laube. *Progress in Movement Pattern Analysis*. BMI Books, 2009.
[37] 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.
[38] F. Mansmann, D. A. Keim, S. C. North, B. Rexroad, and D. Sheleheda. Visual analysis of network traffic for resource planning, interactive monitoring, and interpretation of security threats. *IEEE TVCG*, 13(6):985–997, 2007.
[39] A. Marzuoli, C. Hurter, and E. Feron. Data visualization techniques for airspace flow modeling. In *Proc. Intelligent Data Understanding (CIDU)*, pages 79–86, 2012.
[40] B. McDonnell and N. Elmkvist. Towards utilizing GPUs in information visualization: A model and implementation of image-space operations. *IEEE TVCG*, 15(6):1105–1112, 2009.
[41] Nvidia Inc. CUDA visual computing platform, 2013. www.nvidia.com/object/cuda_home_new.html.
[42] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.
[43] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow map layout. In *Proc. InfoVis*, pages 219–224, 2005.
[44] H. Qu, H. Zhou, and Y. Wu. Controllable and progressive edge clustering for large networks. In *Proc. Graph Drawing*, pages 399–404, 2006.
[45] 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.
[46] D. Selassie, B. Heller, and J. Heer. Divided edge bundling for directional network data. *IEEE TVCG*, 19(12):754–763, 2011.
[47] B. Sridhar, G. Broto Chatterji, and S. Randall Grabbe. Benefits of direct-to tool in national airspace system. *IEEE Trans. Intell. Transport. Sys.*, 1(4):190–198, 2000.
[48] A. Telea. CUDA skeletonization and image processing toolkit, 2013. www.cs.rug.nl/~alext/CUDASKEL.
[49] Thales, Inc. CoFlight Flight Data Processing Framework, 2013. www.thalesgroup.com.
[50] J. Thomas. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society Press, 2005.
[51] C. J. Thompson, S. Hahn, and M. Oskin. Using modern graphics architectures for general-purpose computing: a framework and analysis. In *Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, MICRO 35, pages 306–317, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.
[52] C. Tominski, J. Abello, F. van Ham, and H. Schumann. Fisheye tree views and lenses for graph visualization. In *Proc. Information Visualisation*, pages 202–210, 2006.
[53] J. J. van Wijk. Image based flow visualization. *ACM TOG*, 21(3):745–754, 2002.
[54] N. Wong and S. Carpendale. Supporting interactive graph exploration with edge plucking. In *Proc. IEEE Visualization (interactive posters)*, 2005.
[55] N. Wong and S. Carpendale. Supporting interactive graph exploration using edge plucking. In *Proc. SPIE*, pages 235–246, 2007.
[56] N. Wong, S. Carpendale, and S. Greenberg. Edgelens: An interactive method for managing edge congestion in graphs. In *Proc. IEEE InfoVis*, pages 167–175, 2003.

[57] J. Yi, R. Melton, J. Stasko, and J. Jacko. Dust & magnet: Multivariate information visualization using a magnet metaphor. *J. of Information Visualization*, 4(4):542551, 2006.
[58] H. Zhou, X. Yuan, W. Cui, H. Qu, and B. Chen. Energy-based hierarchical edge clustering of graphs. In *Proc. PacificVis*, pages 55–62, 2008.