Attribute-driven edge bundling for general graphs with applications in trail analysis

Vsevolod Peysakhovich, Christophe Hurter, Alexandru C Telea

Attribute-Driven Edge Bundling for General Graphs with Applications in Trail Analysis

Vsevolod Peysakhovich*
ISAE, Toulouse, France

Christophe Hurter†
DGAC, Toulouse, France

Alexandru Telea‡
University of Groningen, Netherlands

ABSTRACT

Edge bundling methods reduce visual clutter of dense and occluded graphs. However, existing bundling techniques either ignore edge properties such as direction and data attributes, or are otherwise computationally not scalable, which makes them unsuitable for tasks such as exploration of large trajectory datasets. We present a new framework to generate bundled graph layouts according to any numerical edge attributes such as directions, timestamps or weights. We propose a GPU-based implementation linear in number of edges, which makes our algorithm applicable to large datasets. We demonstrate our method with applications in the analysis of aircraft trajectory datasets and eye-movement traces.

Index Terms: I.3.3 [Computing Methodologies]: Computer Graphics—Picture/Image Generation; I.3.6 [Computing Methodologies]: Computer Graphics—Methodology and Techniques

1 INTRODUCTION

Large attributed graphs are ubiquitous in many application domains, such as traffic analysis and planning, network analysis, and bioinformatics. They represent a special subclass of general graphs where edge directions encode essential information for the analysis task at hand. For such graphs, classical visual metaphors such as node-link diagrams produce too much clutter to generate useful pictures.

Graph bundling methods attempt to reduce clutter by grouping edges found to be compatible into so-called bundles. Yet, few such methods handle attributed graphs – edge compatibility is mainly based on spatial position and does not use attributes such as edge direction or data values. Reasoning about such attributes (seeing their values reflected in the bundling) is essential in many applications. Bundling methods for attributed graphs exist, but are computationally not scalable, and thus not usable for large-scale data exploration.

We present *Attribute-Driven Edge Bundling (ADEB)*, a generalized edge bundling technique based on edge advection in compatible vector subspaces, which extends the recent kernel density estimation edge-bundling (KDEEB) [23] with the following contributions:

1. generation of bundled layouts where edge compatibility is defined by one or several numerical edge attributes, *e.g.*, direction, time, or weight;
2. a simple and scalable GPU implementation of our method;
3. applications of ADEB for exploring trail (airline trajectory and eye tracking) datasets demonstrating the method's added value as opposed to classical bundling of undirected graphs.

The structure of this paper is as follows. Section 2 reviews related work on edge bundling and trail visualization. Section 3 presents ADEB's mathematical framework and its implementation. Section 4 applies ADEB to the visual analysis of traffic and eye tracking data. Section 5 discusses our method. Section 6 concludes the paper.

*e-mail: vsevolod.peysakhovich@isae.fr
†e-mail: christophe.hurter@enac.fr
‡e-mail: a.c.telea@rug.nl

2 RELATED WORK

We next briefly review several state-of-the-art edge bundling methods and trail visualization techniques.

2.1 Edge bundling techniques

Key to edge bundling is the reduction of edge-edge clutter by grouping so-called *compatible* edges into spatially compact and thin bundles. This allows an easier visual detection of coarse-scale connectivity patterns in the input graph and supports tasks like finding node-groups which are connected to each other by edge-groups (bundles) [13, 41]. Such strategies are similar to cartographic map generalization, concerned with legibly depicting a complex world in static 2D views [5]. Clutter causes and reduction strategies are discussed by Ellis and Dix [11] and Zhou [45]. Edge bundling has been of increased interest in recent research. We group them according to the edge simplification level (data, geometry, and image), as follows.

Data-based methods: Such methods reduce clutter by showing a simpler graph obtained by filtering or by aggregating edges [11]. Interactive tools, such as NodeTrix, compact dense subgraphs into matrix representations [16, 17]. Ploceus displays networks from different perspectives, at different abstraction levels, and with different edge semantics [33]. Filtering can be done by user-specified queries based on multivariate criteria, or by automated methods using edge metrics such as centrality [43] or spanning trees [32].

Geometry-based methods: Large graphs can be visualized with limited clutter by edge bundling techniques. Edges found to be compatible, *i.e.* close in the graph *drawing* and (optionally) similar in attribute values are deformed to overlap, thereby increasing separating whitespace and thus reducing clutter at the expense of increasing overdraw. Dickerson *et al.* merge edges by reducing non-planar graphs to planar ones [8]. Flow maps use a binary clustering of graph nodes to route curved edges along [36]. Several authors use flow map control-meshes to cluster and route curved edges [38, 46]. Holten generalized this idea by routing edges of compound graphs along the graph's hierarchy drawing [19]. Gansner and Koren bundle edges in a circular node layout by area optimization metrics [14]. Edge routing can also use Delaunay [7] and Voronoi diagrams [28, 29] reflecting the node positions. The most popular (and versatile) bundling methods use force-based techniques that use a force field designed to equal the gradient of a quality function to optimize [10, 20], and have been adapted to separate bundles running in opposite directions [40]. A major issue of geometric techniques is computational complexity. To reduce this, graph optimization techniques and multilevel clustering numerical methods have been proposed [13, 37].

Image-based methods: These methods exploit the parallel computing power of modern GPUs to speed up bundling and/or offer new visual metaphors for bundled graphs. Visual techniques include color interpolation to show edge directions [7, 19]; and mapping attributes such as local edge density or edge lengths via transparency or hue maps [22, 26, 29, 40]. Bundles can be drawn as compact shapes emphasized by shaded cushions [39, 41]. Animated textures can be used to convey edge directions for both static and dynamic graphs [24]. Image-based methods can also massively accelerate bundling: Skeleton-based edge bundling (SBEB) creates strongly ramified bundles by using GPU-computed skeletons or medial axes of the graph drawing's thresholded distance-transform as bundling

cues [12]. Recently, the force-directed idea of [20] was adapted for a GPU setting, offering speed-ups of up to two orders of magnitude compared to state-of-the-art geometric bundling methods [23, 24].

2.2 Trail visualization

Trail datasets consist of sequences of points recording the motion of several objects, *e.g.* vehicles, or tracked trajectories of object parts, *e.g.* eyes or human silhouettes. Such datasets have three main features. First, they can be described as *attributed* graphs, where nodes represent trajectory endpoints and (curved) edge control points represent actual motion paths. Attributes include motion direction, speed, and timestamps. Secondly, they are typically *large*: For instance, 15 minutes of eye tracking data recorded at 50 Hz already gives 45K sample points; air traffic recorded at a country's scale over a week can yield over a million sample points [22]. Finally, they are *complex*, *e.g.* contain many intersecting trails covering large spatial domains. All these factors make trail exploration a challenging proposal with respect to attribute depiction, clutter reduction, and computational scalability. We discuss below several methods related to this task.

Trajectory exploration: Interactive brushing-and-linking techniques help locally reducing clutter when exploring large multivariate trajectory datasets, by selecting up to three dimensions to show in a single 2D view [26]. Focus+context interaction helps further reducing clutter and posing complex spatial-and-data queries [25]. Extracting events of interest from different kinds of spatio-temporal data is discussed (but not fully implemented) by Beard *et al.* [3]. The same query idea is used to extract trajectory parts in [2, 15].

Eye tracking exploration: Density maps tackle the visual scalability problem by aggregating spatially close information for eye track analysis [1, 2, 34]. Separately, Burch *et al.* transform eye tracking data into a dynamic graph to help exploration [4]. Areas of interest (AOI) analysis provides information about dwell times, transitions, fixation points, and AOI hits, but require accurate a priori knowledge to define such areas [9, 18]. A few methods exist that are able to group eye-movement paths; however, they are neither visual nor intuitive to use [31]. An overview of eye movement pattern analysis is given by Laube [30].

Given the size and clutter of trail datasets, bundling methods have emerged as an effective way to show the coarse-scale connectivity structure of these datasets [7, 12, 20, 22, 23, 29]. However, as analyzing trail datasets requires reasoning about trail attributes (*e.g.* timestamps, speed, and direction), undirected bundling techniques are of limited use. For instance, a bundle connecting two node-groups A and B will only tell that A and B are connected, but not in which direction(s) and at which time instant(s) objects moved from A to B. As a side effect, mixing edges of different attribute values in a bundle makes the attribute-based shading-and-blending schemes proposed by several authors [20, 22, 29] of limited effectiveness. Conversely, using bundling techniques for attributed graphs [20, 40] is not practical, as these are far too slow for large datasets such as trail data.

3 ADEB FRAMEWORK

We next describe our ADEB method, a framework that bundles large graphs using an edge-compatibility scheme that can be flexibly controlled to include (mixes of) edge attributes. We first outline the main elements of the KDEEB algorithm from which ADEB is inspired (Sec. 3.1). Section 3.2 presents ADEB's mathematical details. Sections 3.3 and 3.4 explain how to use attributes to create various edge compatibility metrics. Implementation details are given in Sec. 3.5.

3.1 Kernel density bundling

We first review the main principles of the kernel density estimation edge bundling (KDEEB) technique. Let $G = \{e_i\}_{1\le i\le N} \subset \mathbb{R}^2$ be the drawing of a graph of N edges, where each edge e_i is represented by a straight line or arbitrary 2D curve. First, an edge-density map $\rho : \mathbb{R}^2 \to \mathbb{R}^+$ is computed using kernel density estimation

$$\rho(\mathbf{x}\in\mathbb{R}^2) = \frac{1}{h^2}\sum_{i=1}^{N}\int_{\mathbf{y}\in e_i} K\left(\frac{\mathbf{x}-\mathbf{y}}{h}\right), \tag{1}$$

where $K : \mathbb{R}^2 \to \mathbb{R}^+$ is a bivariate radial kernel obtained from a symmetric univariate non-negative kernel function of bandwidth $h > 0$, *e.g.* Gaussian or Epanechnikov. The bundling of G is given by the fixed point of the ordinary differential equation

$$\frac{d\mathbf{x}}{dt} = \frac{h(t)\nabla\rho(\mathbf{x},t)}{\max(\|\nabla\rho(\mathbf{x},t)\|,\varepsilon)}, \quad \forall\mathbf{x}\in G, \tag{2}$$

with initial conditions given by the input graph drawing G. The regularization constant $\varepsilon \ll 1$ takes care of zero gradients. After a few Euler iterations used to solve Eqn. 2, during which one resamples edges, recomputes ρ, decreases h, and makes a 1D Laplacian smoothing pass over G's edges, KDEEB converges and aggregates edges into bundles. Note that KDEEB is nothing else but the well-known mean-shift algorithm [6] applied to the graph drawing G. Hence, KDEEB inherits smoothness, noise robustness, and convergence results proven for mean-shift. For details, we refer to [23].

3.2 Mathematical model for ADEB

We proceed by first introducing a few necessary definitions.

Edge directions: Consider a given drawing $G = \{e_i\}_{1\le i\le N}$ of a directed graph or trail dataset. For each point $\mathbf{x} \in e_i \subset G$, we denote by $\mathbf{d}(\mathbf{x})$ the unit vector tangent to the curve representing e_i, oriented in the direction given by walking on e_i from its start to its end point. For our trail datasets (Sec. 2.2), e_i will be typically a 2D curve rather than straight line (as in a classical node-link graph drawing), so $\mathbf{d}$ changes along e_i. Secondly, note that $\mathbf{d}$ encodes the local directions of the *initial* graph drawing, given as input to the bundling process.

Flow direction map: Given a graph drawing G as above, the *flow direction map* $\theta : \mathbb{R}^2 \to \mathbb{R}^2$ is defined as

$$\theta(\mathbf{x}\in\mathbb{R}^2) = \frac{1}{h^2}\sum_{i=1}^{N}\int_{\mathbf{y}\in e_i} \mathbf{d}(\mathbf{y})K\left(\frac{\mathbf{x}-\mathbf{y}}{h}\right), \tag{3}$$

where K is the same kernel as in Eqn. 1. We can think of θ as being a 2D viscous fluid flow whose boundary conditions are given by the $\mathbf{d}$ values on G. The bandwidth h in this analogy is the fluid viscosity. Note that several points have $\theta(\mathbf{x}) = 0$, *e.g.*, points equidistant from two parallel opposite-direction edges of. In our analogy, these would be flow separation lines. Points $\mathbf{x}$ close to an edge have values $\theta(\mathbf{x})$ close to the edge direction $\mathbf{d}$. The flow magnitude $\|\theta\|$ decreases monotonically and smoothly away from the flow boundary G.

Directional compatibility: At each point $\mathbf{x} \in G$, we define a subspace $\Omega_{\mathbf{x},c}$ of compatible directions as

$$\Omega_{\mathbf{x},c} = \left\{\mathbf{y}\in\mathbb{R}^2\setminus\ker(\theta)\,\middle|\,\frac{\mathbf{d}(\mathbf{x})\cdot\theta(\mathbf{y})}{\|\theta(\mathbf{y})\|}\ge c\right\}\subset\mathbb{R}^2, \tag{4}$$

where $\ker(\theta) = \{\mathbf{x}\in\mathbb{R}^2 | \|\theta(\mathbf{x})\| = 0\}$ and $c \in [-1,1]$ is a compatibility factor. This factor represents the cosine of the maximum allowed angle between the edge direction and flow direction at $\mathbf{y}$. Thus, $\Omega_{\mathbf{x},0}$ covers all points where the flow direction deviates from $\mathbf{d}(\mathbf{x})$ by $\pi/2$ at most, and $\Omega_{\mathbf{x},-1} \equiv \mathbb{R}^2$. In general, given a point $\mathbf{x}$ of the initial graph drawing G, $\Omega_{\mathbf{x},c}$ gives all points in $\mathbb{R}^2$ where the flow map θ has roughly the same direction as $\mathbf{x}$, subject to the factor c.

Given the ρ and θ maps, the subspace $\Omega_{\mathbf{x},c}$ is defined at each point $\mathbf{x} \in G$. We can now define the ADEB bundling of G as the fixed point of the ordinary differential equation

$$\frac{d\mathbf{x}}{dt} = \frac{h(t)\nabla_{\Omega_{\mathbf{x},c}}\rho(\mathbf{x},t)}{\max(\|\nabla_{\Omega_{\mathbf{x},c}}\rho(\mathbf{x},t)\|,\varepsilon)}, \quad \forall\mathbf{x}\in G \tag{5}$$

In contrast to KDEEB (Eqn. 2), the gradient $\nabla_{\Omega_{\mathbf{x},c}}\rho$ is estimated now *only* over the subspace of compatible directions $\Omega_{\mathbf{x},c}$. In analogy with KDEEB, we solve Eqn. 5 by a few Euler iterations, during which we resample edges, recompute ρ and θ, decrease h, and make a 1D Laplacian smoothing pass to remove small wiggles.

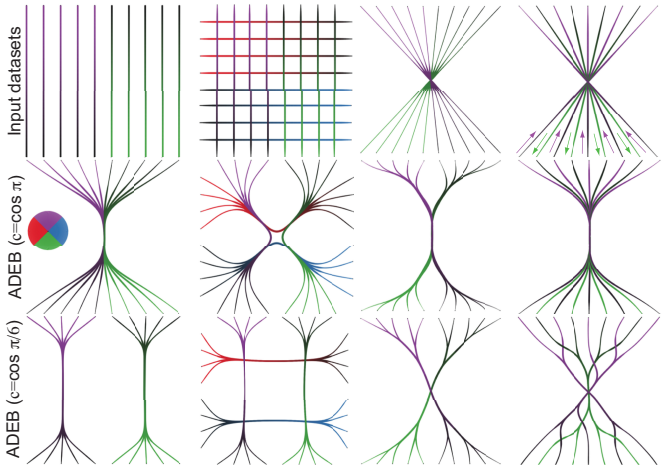


Figure 1: Influence of compatibility factor on bundling result.

In summary, while KDEEB advects points of a given drawing towards local maxima of its edge density, ADEB advects points towards local density maxima of *compatible* directions. Figure 1 shows the influence of the compatibility factor c on four simple graphs. Edges are colored using a directional colormap (see color wheel in figure: edges going west are red, edges going north are purple, edges going south are green, and edges going east are blue). Luminance is linearly increased along edges, so their direction is also shown by following the dark-to-saturated color gradient, in line with [21]. The top row shows the input graphs, which are simple for illustration purposes. The middle row uses $c = \cos(\pi)$, in which case ADEB produces identical results with KDEEB. As shown by the colors, bundles contain edges running in different directions. This creates bundles which arguably do not reflect well the connectivity pattern in the input graph, see e.g. the second row from left. The bottom row shows ADEB for $c = \cos(\pi/6)$, which creates bundles containing only edges having similar directions. As argued also by Selassie *et al.*, this is useful to separate a graph into several bundled 'layers', each representing the bundling of a subset of edges having locally similar directions [40]. The bottom-right image in Fig. 1 shows this: The input graph (top-right image), which contains edges of alternating directions as indicated by the arrows, has been effectively bundled to yield two similar red and purple bundle structures, one for each input direction (compare this example with Fig. 7 in [40]).

3.3 Attribute-based bundling

Apart from direction, we can easily use other edge attributes to define edge compatibility. Consider for instance that edge e_i has, at point $\mathbf{x}$, an attribute value $t_i(\mathbf{x}) \in \mathbb{R}$. We next transform t_i to a vector $\mathbf{t}_i = (\mathbf{t}_i^x, \mathbf{t}_i^y) \in \mathbb{R}^2$ by using polar coordinates, *i.e.* set

$$\mathbf{t}_i^x(\mathbf{x}) = \cos\left(\frac{t_i(\mathbf{x}) - t_{min}}{t_{max} - t_{min}} \cdot \pi\right), \quad \mathbf{t}_i^y(\mathbf{x}) = \sin\left(\frac{t_i(\mathbf{x}) - t_{min}}{t_{max} - t_{min}} \cdot \pi\right) \quad (6)$$

with $t_i \in [t_{min}, t_{max}]$ over the entire G. Next, we use ADEB (Sec. 3.2) unchanged. The resulting flow direction map encodes the compatibility of our edge-attribute t_i. The compatibility factor c is to be set accordingly: For instance, if edges are compatible in time when their timestamp difference is less than σ seconds and the maximum timestamp value is t_{max}, then c should be set to $\cos(\frac{\sigma\pi}{t_{max}})$.

3.4 Multi-criteria bundling

We can use the ADEB framework to bundle graphs using compatibility criteria defined by several attributes. Suppose that we calculated two flow direction maps corresponding to edge direction and edge timestamps respectively. We can next compute two subspaces of compatible directions and compatible timestamps. Next, by computing the gradient $\nabla\rho$ over the *intersection* of these two subspaces, the result given by Eqn. 5 is a multi-criteria bundled layout.

Figure 2 shows this for a dataset recording the gaze for a one-minute exploration of Ilya Repin's painting *Unexpected Visitor*. Similar to Yarbus [44], a subject was asked to look at the painting and first assess and verbalize the age of the main characters, and next the age of secondary ones. Using time-based compatibility, ADEB successfully bundles together the trails separated in time (Fig. 2 b, trails colored by time via a blue-green-red colormap). We find that there are two main 'visual scans' in this scenario – first, a scan between the faces of foreground characters A and B (green bundle), followed by a second scan between background characters C and D (red bundle). However, since we don't use direction for bundling, we cannot tell if in these scans the eye moved from left to right, right to left, or both directions. Figure 2 b shows the result of directional bundling, colored by trail directions like in Fig. 1. We now see that both scans A-B and C-D include left-to-right (blue) and right-to-left (red) bundles of similar thickness. Hence, the eye moved in both cases in *both* directions, and it did so in a balanced way (comparable amount of times in both directions). However, close to character C, the ends of the C-D bundles are pulled towards and mixed with the A-B bundles, even though the two scans are temporally separated. Using time-and-direction based compatibility, we can obtain both desirable effects shown before (Fig. 2 d): Each scan is separated in a left-to-right and right-to-left bundle, and temporally different scans are not mixed.

3.5 Implementation details

We evaluate the two components θ_x and θ_y of the flow direction θ by using Eqn. 3 twice, for the two corresponding components $\mathbf{d}_x$ and $\mathbf{d}_y$ of the edge direction vector $\mathbf{d}$ respectively. We implement this on the GPU by splatting the kernel K at all edge sample points using additive blending, and accumulating the resulting θ_x and θ_y values in two floating-point textures. Next, we compute the density gradient $\nabla_{\Omega_{\mathbf{x},c}}\rho$ by finite differences, on a compatible subspace $\Omega_{\mathbf{x},c} \cap \nu_\varepsilon(\mathbf{x})$, where $\nu_\varepsilon(\mathbf{x})$ is a neighborhood of $\mathbf{x}$ of radius $\varepsilon \geq h > 0$. While covering the neighborhood $\nu_\varepsilon(\mathbf{x})$, each point $\mathbf{y}$ of it is tested for directional compatibility. The finite differences are computed only on points that pass the compatibility test in Eqn. 4. Finally, while doing the bundling iterations that solve Eqn. 5, we relax the user-given compatibility value c so it reaches $\cos\pi$ at the last iteration. This way, the first iterations force edges to aggregate into compatible bundles, while the last iterations 'compact' edges already found to be compatible into smoother final bundles.

Our method has similar complexity and computational performance with KDEEB. Specifically, computing the flow direction map θ takes twice the time required for KDEEB's density map computation. Estimating the directional compatibility criterion adds only a simple logical condition for the density gradient computation. Memory-wise, our method requires two floating-point buffers of the size of the graph drawing G, which is a negligible cost increase as compared to KDEEB. Compared to FDEB [20] and divided edges [40], two other prominent directional bundling techniques, our method is over one order of magnitude faster (see also Sec. 5).

We render the bundled edges as curves colored by the value of one attribute of interest, *e.g.* time or direction. We also scale the curve thickness at each point $\mathbf{x}$ with the local edge-density value $\rho(\mathbf{x})$ (Eqn. 1). Bundles (or bundle fragments) thus appear locally thicker in areas where they contain many edges, thereby enabling users to estimate the importance of the respective connection patterns.

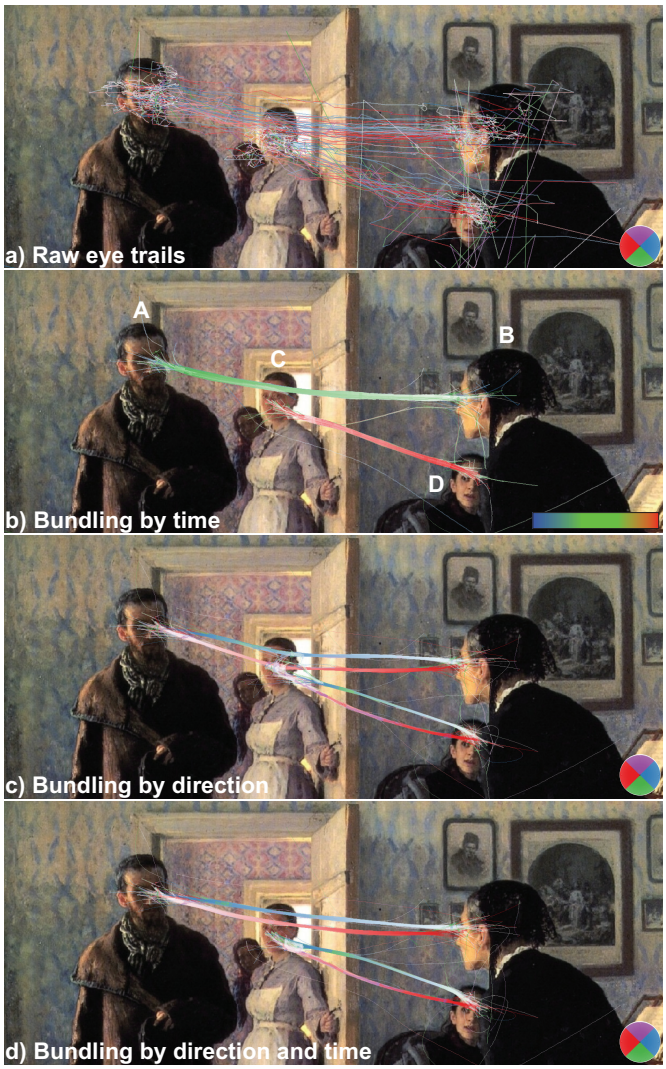


Figure 2: Multi-criteria bundling of an eye tracking dataset.

4 Applications

We next present two applications of ADEB for the qualitative analysis of trail datasets represented by directed attributed graphs.

4.1 Aircraft traffic exploration

Our first application concerns the exploration of aircraft traffic. Given a geographical region (*e.g.*, a country), aircraft position is periodically recorded over a given time interval (*e.g.*, days or weeks) by a mix of observations from ground radar stations and signals emitted by onboard transceivers. This generates a set of trails, attributed by the measured quantities (aircraft timestamp, position, speed, height, and ID). Considering landing and takeoff points as nodes and trails as edges, such data can be modeled as a directed attributed graph. Such data is used by air traffic controllers both online (to keep safe distances between flying aircraft) and offline (to optimize air traffic flows, *e.g.*, minimize delays and optimize flight routes).

Edge bundling has already been used to highlight coarse-scale connection patterns in air traffic data [23, 22]. However, this causes problems when tasks at hand involve reasoning about flight *directions*. Indeed, existing directional bundling techniques are either not scalable enough [40] or need expensive preprocessing steps [12]. To separate different directions mixed in the same bundle, interaction can be used, *e.g.* the MoleView semantic lens that smoothly interpolates between bundled and unbundled trails at locations of interest [25]. However, this approach only provides local insights.

Dataset: The analyzed dataset is 24 hours (February 8^{th}, 2009) of recorded aircraft trails over France (Fig. 3, 18858 trajectories, 415257 records or sample points). We detail below two exploration scenarios, performed by an ATC expert, using our ADEB bundling technique, embedded in the FromDaDy visual exploration tool [26]. Findings have been cross-checked by a separate air traffic controller.

Overview analysis: Exploration started by analyzing an overview of the flight patterns. Figure 3 a shows the input trail set. Trails are color-coded based on direction, as in Fig. 2. To emphasize direction, we made them dark and saturated at the start points and bright and desaturated at end points. Figure 3 b shows the directionally bundled trails, color-coded as above. Compared to the unbundled image, we can now see the different incoming and outgoing aircraft 'flows' at several main European cities (Paris, London, Zürich). We'll explore these further below. Secondly, we notice the complex flight pattern over Lyon, which differs significantly from the pattern above Paris. Indeed, while Paris does not have transit traffic (aircrafts that fly over a given area without landing), Lyon has major transit flows. One such example is formed by the red and blue bundles linking Zürich to Spain (Fig. 3 b, bundle *b*). Over the entire image, we see several such bundle-pairs having similar thickness in the two directions, *e.g.* the one ending in London (Fig. 3 b, bundle *a*). Since bundle width maps the number of bundled trails (Sec. 3.5), this shows us that such paired flows contain balanced traffic in the two directions. Such flows correspond to daily commuting flights.

Detailed analysis: We next zoom to explore flight details in a circle of about 100 km around the Paris area (Fig. 3 c). We discover that this flow pattern has four main incoming and four outgoing flows (see arrows in the figure). Also, we see that there are only a few thin bundles whose endpoints are not in the center of the image – thus, we confirm the overview finding that Paris has little transit traffic. Most bundles start/end at two concentrated locations (dotted circles in the figure). These correspond to the two main Paris airports, Roissy and Orly. The detail analysis also shows us facts that were not visible on the overview: For Roissy, the thickness of the main bundles (marked by arrows) show that west and north departures are far fewer than south and east ones. More interestingly, we see that, at this scale, bundle-pairs corresponding to departure-arrival flows in the same main direction, are far less parallel than in the coarse-scale visualization. This matches existing knowledge that landing-departing patterns close to airports are much wider spread (in direction) than in-flight patterns, as the former need all to share a smaller geographical area. For Orly airport, we immediately see that traffic is far smaller than for Roissy, and it goes mainly to the south and south-west (bundles *sw* and *s*). Only a thin bundle goes from Orly to the north (arrow *n*). This matches known evidence on this airport: Orly mainly serves domestic flights and is in the north of France; as such, its traffic goes mainly to the south and south-west.

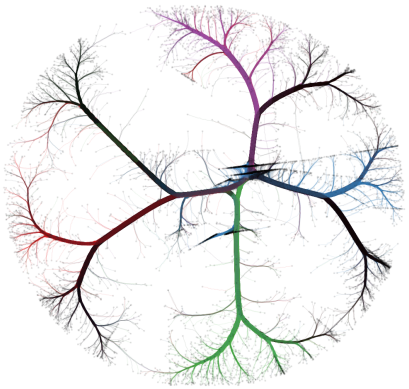


Figure 4: Aircraft traffic over Paris (bundled with KDEEB). Compare with the ADEB directional bundling in Fig. 3 c.

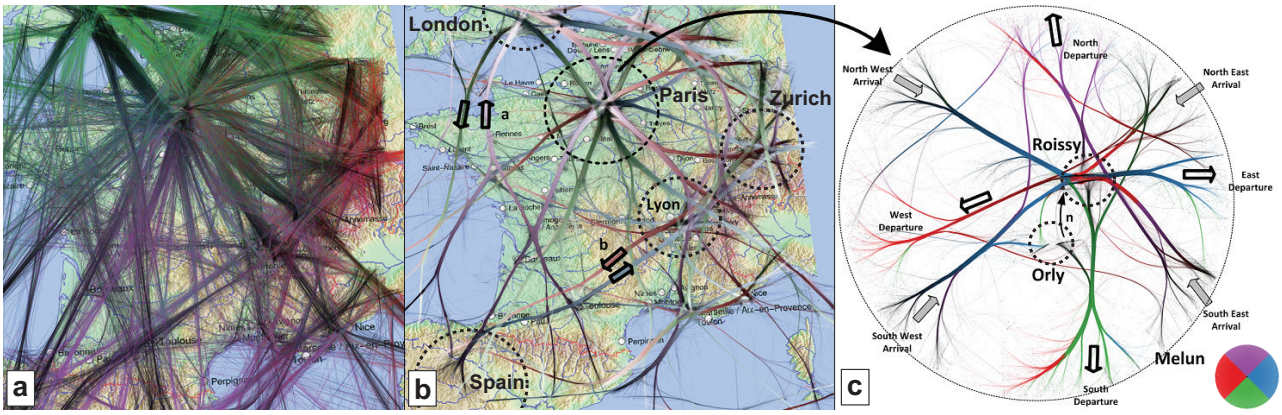


Figure 3: Aircraft trails analysis. (a) Raw data. (b) Directional bundling over France. (c) Zoom-in over Paris area.

Such insights cannot be obtained with a direction-agnostic bundling method. To illustrate this, Fig. 4 shows the same zoom-in as in Fig. 3 c, bundled with KDEEB. The north departure bundle is now mixed with north-east arrivals, and east departures get mixed with south-east arrivals. Also, the several thin traffic bundles (aircraft that do not start/stop at Paris) visible in Fig. 3 c get now aggregated. In other words, undirected bundling keeps the very-coarse connection patterns, but eliminates the finer ones that depend on direction.

4.2 Eye tracking exploration

Recent eye-tracking technology developments made eye movement data increasingly used in applications in neuroscience, psychology, human factors, training, and marketing. Such data consists of sampled trails giving the motion of the subject's gaze while completing a given task. Its two most important elements are *fixations* and *saccades*. Fixations describe points that the eye is attracted to (thus, where attention is drawn). Saccades link fixations and model how one 'scans' the image to link information given by fixations.

For fixation events, studied more often, attention and saliency maps are often used to show regions of interest by locating high-density fixation areas. However, the *saccades* linking such points are also of great importance, especially to understand how low-level fixation information is aggregated to form a higher-level understanding of the picture. Specifically, spatially and temporally close saccades are to be grouped into so-called *scanpaths*, and further analyzed to understand how the image was processed [44]. Two kinds of visualizations exist for scanpaths: either the gaze is dynamically played back, or the sequence of saccades is depicted statically by connecting consecutive fixation points by arrows. Both visualizations are quite inconvenient for pattern analysis. Playback is necessarily local in both time and space, and thereby cannot show the grouping. Playback is also time-consuming. Static drawings of raw scanpaths generate cluttered images, especially when analyzing data produced by lengthy experiments containing thousands of saccades [4, 27].

We next describe the use of attribute-based bundling for analyzing such eye tracking datasets. First, we outline how we extracted an attributed graph from raw eye-tracking data. Next, we detail two experiments where our bundling proved to be an effective aid for getting insight into the recorded data (Secs. 4.2.1 and 4.2.2).

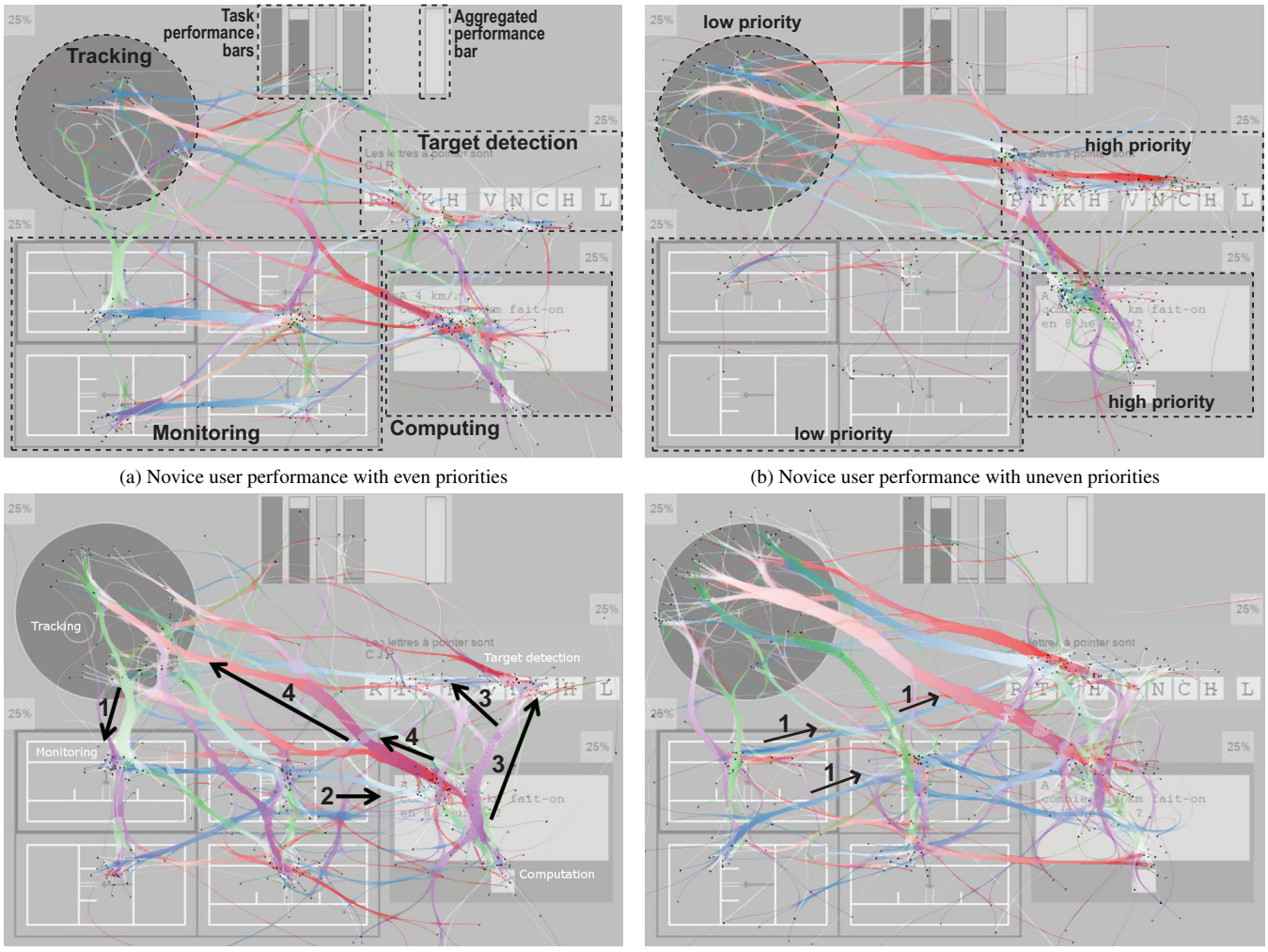
Preprocessing: Raw gaze data, recorded at 50 Hz, was preprocessed to extract fixations and saccades as follows. Consecutive sample points located in a square of 20×20 pixels and separated by at least 200 ms were considered to be a fixation event. Points marked as a one fixation event were replaced by their average. This reduces small-scale noise (scattered close fixation points) generated by micro-saccades and eye tracking device imprecision during a fixation. Clustering algorithms could be used [6] to further reduce clutter and aggregate adjacent fixations corresponding to one object of interest but separated spatially due to the imprecision of human visual system. Next, trails formed by consecutive sample points were cut between fixations, yielding the saccades. Finally, a graph was created using fixations as nodes and saccades as curved edges. Note that since human eye does not encode any information during the saccadic movement, curved edges of our graph do not have direct attentional interpretation and thus could be distorted.

4.2.1 Multitask experiment

Eye tracking technology can be used to study repartition of visual attention. Consider, for example, how a pilot watches the cockpit instruments during a certain maneuvre (task). Such tasks, *e.g.* take-off or landing, are typically split into multiple subtasks, each involving several instruments between which the pilot has to allocate attention. For each subtask, instruments have to be watched in appropriate order and with a given frequency. For instance, the priority of the Primary Flight Display (PFD) instrument with respect to Flight Control Unit (FCU) will be different for the final landing approach as compared to the cruise phase [35].

Data and tasks: This dataset is part of the priority management testing of the pilot selection at French Civil Aviation School [35]. The subject has to perform a main task containing four concurrent subtasks of different nature (tracking, monitoring, target detection, mental computing), with two priority conditions: (a) all subtasks have even priorities, and (b) two high-priority and two low-priority subtasks (uneven priorities). Tasks are performed using a dashboard-like multitask interface on which several instruments show dynamically-changing data. The interface and location of the various instruments used for each task is shown in the background of Fig. 5a. The top bar widgets let subjects monitor their performance (four bars for each task, and one aggregated-performance bar). For the uneven condition, subtask priorities are shown in Fig. 5b. A detailed description of this experiment and the multitask design are given in [35]. Two subject groups, one formed by experienced pilots, and the other one formed by novices, performed the task under both priority conditions, while their gaze was continuously recorded. For each such run, which took 4 minutes, we used the described above preprocessing to obtain a graph having 234 saccades of 24 samples each on average. Using this data, we next try to find similarities and differences in task execution across users and priority conditions.

Results: Figure 5 shows the result of applying directional bundling on four sample test runs, involving both priority conditions and user groups, colored by direction as in earlier figures. Directional bundling allows us to identify the main scanpaths (characteristic sequences of saccades) involved in each run, and also to compare these scanpaths across users and conditions. Data is normalized in the sense that all runs are of the same length (in time and number of sample points), and also contain similar numbers of fixation points. As such, differences between such bundled images reveal both inter-subject and intra-subject similarities and differences.



(a) Novice user performance with even priorities

(b) Novice user performance with uneven priorities

(c) Expert performance with even priorities

(d) Expert performance with uneven priorities

Figure 5: Bundled eye-tracking trails of novice and expert subjects for a multitask experiment done with two priority conditions.

Let us first analyze the influence of expertise level on the visual attention distribution. The expert has more pronounced transitions between the four subtasks as compared to the novice (Figs. 5a, 5b), as shown by the thicker bundles in Figs. 5c, 5d. Given that the average saccade speed is similar over humans, and the experiment duration is identical for both user types, we deduce that the expert needs less time to perform a subtask and spends more time to switch between the four screen areas to update his knowledge about subtasks. He also largely ignores the performance bar in his 'attentional walk' as compared to a novice. This way of scanning an image, also called *ocular strategy*, leads to better performance [35].

The ocular strategy changes when the priorities of the four subtasks are uneven, as shown by the different structure of the bundles in Figs. 5b,5d as compared to Figs. 5a,5c. Yet, the priority condition affects novices and experts differently. The expert shifts his attention to the more important subtasks, as shown by the increased bundle coverage in the right part of Fig. 5d. In contrast, the novice almost abandons the monitoring task and spends most of the time on the two high-priority subtasks. For the uneven priority case, the tracking task induces significant attention for both the expert and novice. This can be due to the fact that, in this experiment, the user's right hand always rests on the joystick that controls the tracking subtask, so they unconsciously dedicate significant attention to this task, even when they are instructed that tracking is a low-priority subtask.

Directional bundling also shows us the sequence (order) of attendance of different instruments. Consider the expert participant with even priorities (Fig. 5c). The main transitions between subtasks, covering about 80% of the entire set of saccades, are captured by the following thick bundles: tracking to monitoring (green bundle 1), monitoring to computing (two merging blue bundles 2), computing to target detection (purple bundle 3), computing to tracking (red bundle 4). Hence, a special order of subtask monitoring can be deduced. First, we deduce that the expert monitored information in anticlockwise order (tracking, then monitoring, then computing, then tracking again). We also see, for the even priority case, that neither the expert nor the novice do have significant transitions between target detection and monitoring. For the uneven priority case, however, the expert has many transitions from monitoring to target detection (Fig. 5d, blue bundles 1). This shows that the expert remembered that target detection is of high priority, and periodically switches attention to it from the less important monitoring.

4.2.2 Landing experiment

Data and tasks: This dataset is part of a study conducted at the French Aerospace Engineering School. During the experiment, a pilot performed a landing manoeuvre in a flight simulator. The experiment lasted 15 minutes. The aim was to test a new cockpit instrument providing landing assistance, which we next call the Landing Aid Instrument (LAI). More specifically, we wanted to understand whether (and how) the new LAI is used along the other instruments.

Eye tracking recording resulted in a graph with 1194 saccades of 8 samples each on average.

Results: Figure 6 shows the raw trails from this experiment, the results of our ADEB bundling, and those of undirected (KDEEB) bundling. Bundles are directionally colored, as in earlier figures. The background image shows the cockpit dashboard. Carefully inspecting the raw trails (Fig. 6 a) reveals several saccades connecting the Primary Flight Display (PFD, left), Navigation Display (ND, center), Flight Control Unit (FCU, top-right), and the new Landing Aids Instrument (LAI, bottom-right). However, the raw trails display is too cluttered. We cannot be sure that we found all relevant instrument-to-instrument trails, and we cannot see high-level scanpath patterns. When using ADEB bundling (Fig. 6 b), the following connections become obvious: faint purple bundle 1 LAI→FCU, blue bundle 2 PFD→LAI passing by ND, red bundle 3 in the opposite direction, red bundle 4 FCU→PFD, and blue bundle 5 in the opposite direction. The bundle PFD-ND is denser than the PFD-FCU one, and much denser than the PFD-LAI one. Thus, the main pattern in the pilot's gaze can be deduced: PFD→ND→FCU→PFD. The LAI is rarely consulted because of its novelty – the pilot is used to land the aircraft without this instrument. In more detail, we see that there is no dense connection FCU→LAI. The pilots explained that, after entering flight parameters on the FCU, they are used to check them immediately on the PFD (as shown by the red bundle 4). Thus, to better integrate the new LAI in this maneuvre, a new training protocol should be designed in which pilots are specifically told when to add the visual consultation of the LAI in their action sequence.

Figure 6 c shows the results of undirected bundling. We see, as in our earlier aircraft trail-analysis (Fig. 3 c *vs* Fig. 4), that undirected bundling highlights only coarse-level connection patterns and smooths our finer-grained ones. For example, the red LAI→PFD and FCU→PFD bundles visible with ADEB (Fig. 6 b) are now hard to see, as they appear to end at ND. More problematically, we notice a quite thick bundle between FCU and LAI (Fig. 6 c, bundle 1), which suggests that the subject *did* in fact frequently use the LAI in conjunction with the FCU, which we know is not true. ADEB bundling does not show this bundle, as there are not enough saccades coherent in *both* space and time between FCU and LAI, and thus conveys us the correct insight (LAI rarely used).

5 Discussion

We discuss next several technical aspects of our method.

Generality: Most earlier bundling methods use only edge length, absolute angles, and relative distances (the latter which we also use). So far, only [40] explicitly shows directional bundling; [20] mentions this possibility, but does not actually demonstrate it. Separately, attribute-based bundling is noted as a possibility in [41], but not actually demonstrated. In contrast to all above, our ADEB demonstrates how both local edge direction and edge attributes can be added to earlier geometric compatibility metrics.

Scalability: The most prominent bundling techniques using edge compatibility measures (beyond edge distance) are [20, 40]. Both have a complexity of $O(E^2C)$ where E is the number of edges in the input graph, and C the number of edge sampling-points. In contrast, our method, just as KDEEB [23], is $O(C)$. This makes our method considerably more scalable than [20, 40], and of about the same scalability as [23]. In detail, FDEB takes 19 seconds on a graph of 2101 edges and 205 nodes (the well-known *US airlines* dataset), with 12 sample-points/edge on average (thus, roughly 25K sample points in total); the method in [40] takes 24 seconds for the same graph, for a sampling of up to 25 points/edge (up to 75K sample points), excluding preprocessing costs which authors note to be 10% of the total cost. For the same graph, KDEEB takes 0.5 seconds, at 86K sample points. ADEB takes 1.3 seconds for the same amount of sampling points. Overall, ADEB is roughly twice as slow as the undirected KDEEB, but about 18 times faster than the directional method in [40].

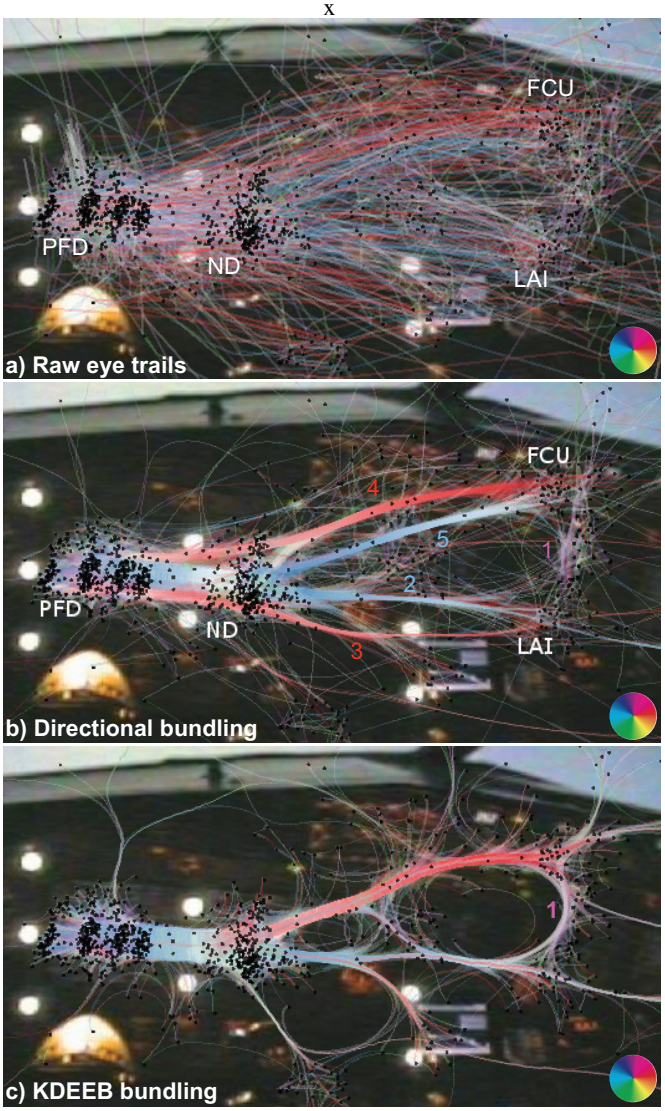


Figure 6: Raw and bundled eye trails, landing scenario (Sec. 4.2.2).

Interaction: Edge bundling aims to create simplified *overviews* of complex graphs. To solve specific tasks, extra tools such *e.g.* local levels-of-detail [25], attribute-based selection [26], or interactive aggregation [42], are obviously beneficial. Such tools can be easily added to ADEB. In line with earlier bundling papers [7, 12, 20, 23, 29, 40], we do not detail such extensions here, so that our contribution on directional bundling is more focused and easier to separate from such additional tooling mechanisms.

Limitations: Our attribute-based bundling implies that we can map distances between attribute-values (in their original space) to angle distances (Eqn. 6). This is immediate for quantitative and ordinal attributes, but is not evident for categorical attributes. For the latter case, such a mapping would require an algebra definition on the category space. Hence, for categorical attributes, we suggest to cluster trails according to their category and perform separate bundling computations for each such cluster (much like [41]). Separately, we showed the added-value of ADEB *vs* a single undirected bundling method (KDEEB). This leaves the open question whether *other* undirected bundling methods would be comparably better. While such

comparisons are yet to be done, we argue that, for tasks that require seeing and analyzing bundles of different directions, all undirected-bundling methods would exhibit similar limitations as KDEEB. Finally, we note that our use-cases of ADEB for analyzing eye-tracking data do not imply that ADEB is the ultimate technique to gain all types of insights from such data, as compared to *e.g.* areas of interest (AOIs), dwell-time-per-AOI, or transition matrices. Rather, we employ this use-case to show how ADEB can be of added value as opposed to a raw or undirected-bundling display of trails (Fig. 6).

6 Conclusion

In this paper, we proposed ADEB, a method that creates edge bundles from attributed graphs where edge compatibility can be defined by one or several attributes. ADEB is fast, simple to implement, scalable, generic, and could be easily extended to handle dynamic graphs. We demonstrated ADEB, and implicitly the added-value of directional bundling, by showing two applications in the analysis of aircraft trails and eye movement recordings.

Future work can target several aspects. Refined compatibility metrics (based on graph attributes) and visual metaphors could allow users to better spot specific patterns of interest in large eye-tracking datasets. Secondly, ADEB could be adapted to target additional domains, such as creating simplified attribute-based visual representations of DTI fiber tracts. Finally, we aim to explore how ADEB can be extended to handle attribute-based bundling of dynamic graphs.

Acknowledgements

The authors thank Nadine Matton, Frederic Dehais and Sebastien Scannella for providing gaze datasets.

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 Info Sci*, 25(9):1347–1370, 2011.

[3] K. Beard, H. Deese, and N. Pettigrew. A framework for visualization and exploration of events. *J. Inf. Vis.*, 7(2):133–151, 2008.

[4] M. Burch, F. Beck, M. Raschke, T. Blascheck, and D. Weiskopf. A dynamic graph visualization perspective on eye movement data. In *Proc. ETRA*, pages 151–158. ACM, 2014.

[5] B. Buttenfield and R. McMaster. *Map Generalization: Making rules for knowledge representation*. J. Wiley & Sons, 1991.

[6] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603–619, 2002.

[7] 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.

[8] M. Dickerson, D. Eppstein, M. Goodrich, and J. Meng. Confluent drawings: Visualizing non-planar diagrams in a planar way. *J. Graph Alg Appl*, 9(1):31–52, 2005.

[9] A. Duchowski. *Eye tracking methodology: Theory and practice*, volume 373. Springer, 2007.

[10] T. Dwyer, K. Marriott, and M. Wybrow. Integrating edge routing into force-directed layout. In *Proc. Graph Drawing*, pages 8–19, 2007.

[11] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE TVCG*, 13(6):1216–1223, 2007.

[12] 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.

[13] E. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. In *Proc. PacificVis*, pages 187–194, 2011.

[14] E. Gansner and Y. Koren. Improved circular layouts. In *Proc. Graph Drawing*, pages 386–398, 2006.

[15] R. Güting and M. Schneider. *Moving objects databases*. Elsevier, 2005.

[16] S. Hadlak, H. Schulz, and H. Schumann. In situ exploration of large dynamic networks. *IEEE TVCG*, 17(12):2334–2343, 2011.

[17] N. Henry, J. Fekete, and M. J. McGuffin. Nodetrix: a hybrid visualization of social networks. *IEEE TVCG*, 13(6):1302–1309, 2007.

[18] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, and J. Van de Weijer. *Eye tracking: A comprehensive guide to methods and measures*. Oxford University Press, 2011.

[19] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE TVCG*, 12(5):741–748, 2006.

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

[21] D. Holten and J. J. van Wijk. A user study on visualizing directed edges in graphs. In *Proc. ACM CHI*, pages 2299–2308, 2009.

[22] C. Hurter, O. Ersoy, S. Fabrikant, T. Klein, and A. Telea. Bundled visualization of dynamic graph and trail data. *IEEE TVCG*, 20(8):1141–1157, 2014.

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

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

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

[26] 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.

[27] H. Jarodzka, K. Holmqvist, and M. Nyström. A vector-based, multidimensional scanpath similarity measure. In *Proc. ETRA*, pages 211–218. ACM, 2010.

[28] A. Lambert, R. Bourqui, and D. Auber. 3D edge bundling for geographical data visualization. In *Proc. IV*, pages 329–335, 2010.

[29] A. Lambert, R. Bourqui, and D. Auber. Winding roads: Routing edges into bundles. *CGF*, 29(3):432–439, 2010.

[30] P. Laube. *Progress in Movement Pattern Analysis*. BMI Books, 2009.

[31] O. Le Meur and T. Baccino. Methods for comparing scanpaths and saliency maps: strengths and weaknesses. *Behavior research methods*, 45(1):251–266, 2013.

[32] B. Lee, C. S. Parr, C. Plaisant, B. B. Bederson, V. D. Veksler, W. D. Gray, and C. Kotfila. Treeplus: Interactive exploration of networks with enhanced tree layouts. *IEEE TVCG*, 12(6):1414–1426, 2006.

[33] Z. Liu, S. B. Navathe, and J. T. Stasko. Network-based visual analysis of tabular data. In *Proc. VAST*, pages 41–50. IEEE, 2011.

[34] A. Marzuoli, C. Hurter, and E. Feron. Data visualization techniques for airspace flow modeling. In *Proc. CIDU*, pages 79–86. IEEE, 2012.

[35] N. Matton, P. Paubel, J. Cegarra, and E. Raufaste. Resource allocation strategies in multitasking after switch in task priorities. *Advances in Cognitive Engineering and Neuroergonomics*, 11:187, 2014.

[36] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow map layout. In *Proc. InfoVis*, pages 219–224, 2005.

[37] S. Pupyrev, L. Nachmanson, S. Bereg, and A. E. Holroyd. Edge routing with ordered bundles. In *Graph Drawing*, pages 136–147, 2012.

[38] H. Qu, H. Zhou, and Y. Wu. Controllable and progressive edge clustering for large networks. In *Proc. Graph Drawing*, pages 399–404, 2006.

[39] 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.

[40] D. Selassie, B. Heller, and J. Heer. Divided edge bundling for directional network data. *IEEE TVCG*, 19(12):754–763, 2011.

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

[42] S. van den Elzen and J. van Wijk. Multivariate network exploration and presentation: from detail to overview via selections and aggregations. *IEEE TVCG*, 20(12):2310–2319, 2014.

[43] F. van Ham and M. Wattenberg. Centrality based visualization of small world graphs. *CGF*, 27(3):975–982, 2008.

[44] A. L. Yarbus. *Eye Movements and Vision*. Plenum Press, 1967.

[45] H. Zhou, P. Xu, Y. Xiaoru, and Q. Huamin. Edge bundling in information visualization. *Tsinghua Sci. Tech.*, 18(2):148–156, 2013.

[46] 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.