Functional Decomposition for Bundled Simplification of Trail Sets

Christophe Hurter, Stéphane Puechmorel, Florence Nicol, Alexandru C Telea

Functional Decomposition for Bundled Simplification of Trail Sets

Christophe Hurter, Stéphane Puechmorel, Florence Nicol, Alexandru Telea

Abstract—Bundling visually aggregates curves to reduce clutter and help finding important patterns in trail-sets or graph drawings. We propose a new approach to bundling based on functional decomposition of the underling dataset. We recover the functional nature of the curves by representing them as linear combinations of piecewise-polynomial basis functions with associated expansion coefficients. Next, we express all curves in a given cluster in terms of a centroid curve and a complementary term, via a set of so-called principal component functions. Based on the above, we propose a two-fold contribution: First, we use cluster centroids to design a new bundling method for 2D and 3D curve-sets. Secondly, we deform the cluster centroids and generate new curves along them, which enables us to modify the underlying data in a statistically-controlled way via its simplified (bundled) view. We demonstrate our method by applications on real-world 2D and 3D datasets for graph bundling, trajectory analysis, and vector field and tensor field visualization.

Index Terms—path visualization, trajectory visualization, edge bundles, functional decomposition, path generation, streamlines

1 Introduction

Bundling techniques reduce visual clutter in large graph drawings or trail-sets by grouping curves into bundles. Clutter is traded for overdraw, making the coarse-scale patterns in the input data better visible. In the last decade, tens of different bundling techniques have been proposed for the visual exploration of graphs, networks, motion data, fiber sets, and multidimensional data. Challenges such as computational scalability [32,41,66], incorporating data attributes into bundling [50], shaded visualization of bundles [39,63], and handling dynamic data [33,47] have been extensively covered by existing techniques.

Bundling can be seen as an *image-space* simplification operator [42] that creates a sparse, easy to explore, representation of a dense curve drawing. Bundles can be seen as simplified proxies for sets of original data elements. In many cases, such curves are actually sampled paths of some processes that have a functional nature. Hence, three questions arise: (1) How can we model curves by recovering their underlying functional nature? (2) How can we model the *variability* of data elements that a bundle summarizes? (3) How can we *manipulate* the original data by manipulating its simplified (bundled) view? To our knowledge, no work has jointly addressed these three questions yet.

We approach these questions by using proven statistical tools from functional data analysis (FDA). In contrast to current methods, we model bundling as a *data-variability* representation and simplification process. We consider input curves as large sequences of possibly noisy spatial sample points. To answer (1), we model such curves as finite linear combinations of spline basis functions. This has two key advantages: It reduces the number of sampling points required to represent complex-shaped trails, and it allows us to directly control the desired smoothness of our bundles. For instance, complex-shaped aircraft trails require thousands of sample points, but can be efficiently encoded with fewer basis functions and associated coefficients. To answer (2) and (3), we use Functional Principal Component Analysis (FPCA), a well-known statistical tool in FDA, that generalizes multivariate principal component analysis to a functional setting. Given a group of similar curves, FPCA models any curve in the group as a sum of the mean curve and a linear combination of so-called principal component functions. FPCA has two major advantages. First, the PCF's capture the shapes of the curve sample with a small number of components. Secondly, the PCF expansion coefficients, called scores, reveal shape differences and variability of each curve around the centroid. As curves and scores correspond one-to-one, we can easily generate arbitrary curves that are similarly spatially spread, and have similar shapes, to the bundled ones, by sampling new scores from the estimated score densities. This enables us to easily modify a centroid curve and next generate an arbitrary number of curves that (a) follow the modified centroid at a user-controlled distance but (b) follow the spatial variability and shape of the input curves. This process, which we dub 'unbundling', allows us to explore and manipulate dense datasets via their bundled versions.

- *C. Hurter, S. Puechmorel, and F. Nicol are with ENAC. E-mail: {christophe.hurter | stephane.puechmorel | florence.nicol}@enac.fr.*
- *Alexandru Telea is with the University of Groningen: E-mail:a.c.telea@rug.nl.*

2 Notations and Related Work

Following Lhuillier *et al.* [42], let $P = \{\gamma_i\}$ be a set of paths embedded in $\mathbb{R}^d$, where d typically is 2 or 3. These can be either edges in a graph drawing [28] or curves in a trail-set [50,59]. The bundling $B(\gamma_i)$ of a path γ_i then satisfies

$$\forall(\gamma_i,\gamma_j) \in P \times P | \kappa(\gamma_i,\gamma_j) < \kappa_{max} \rightarrow \delta(B(\gamma_i),B(\gamma_j)) \ll \delta(\gamma_i,\gamma_j). \quad (1)$$

Here, δ is a distance metric between $\mathbb{R}^d$ curves, *e.g.*, Hausdorff distance [19]; and κ reflects the dissimilarity of two paths γ_i and γ_j, which always considers the distance δ between paths, but can also consider path type, weights, timestamps, or directions [31,63]. Simply put, Eqn. 1 says that the bundled versions of two sufficiently similar curves are much closer than their unbundled versions. Bundling methods are usually grouped into geometry-based *vs* image-based [30,74]. In our context it is more useful to discuss bundling methods related to how they address the three questions outlined in Sec. 1, as follows.

Q1. Functional modeling: When the input data P can be modeled as *functions*, it is advisable to use tools that preserve this model. Functional Data Analysis (FDA) [56] does precisely this: While in traditional multivariate statistics one samples functions at several points, ending up with a finite-dimensional vector representation, FDA expands the functional data on a basis, then uses the vector of expansion coefficients in a multivariate framework. A clever choice of the expansion basis allows directly capturing important data features such as smoothness and details. However, FDA was mainly used for real-valued functions only, *e.g.* spectrometric data [58] or weather data [35], little was done for vector-valued functions, such as our 2D or 3D paths. The main issue for vector-valued functions is the interdependence of coordinates that, as we shall see, precludes the analysis in a per-coordinate fashion.

Q2. Bundle modeling: Eqn. 1 tells that data-similar paths ($\kappa(\gamma_i,\gamma_j) < \kappa_{max}$) get bundled into spatially very close curves $B(\gamma)$. Bundles are typically not modeled *explicitly* but emerge purely as visually separate shapes in the final image. It is thus important to control which paths γ_i end in a given bundle. For this, two kinds of methods exist. *Explicit* methods build a support data structure I from P and next define B based

on I: Hierarchical methods build I from the hierarchy of a compound graph [10,13,23,28,68] or a tree extracted from a DAG [54]; general-graph methods use spanning trees [51], Steiner trees [9], spiral trees [8], Voronoi diagrams [14,39], and Delaunay triangulations [14,55]. Given I, paths are routed along it as smooth curves, *e.g.* B-splines [10,23,28,52,68]; Bézier splines [5]; NURBS [55]; and cubic curves [25]. Explicit methods control which paths end in the same bundle, *i.e.*, how the input P is *partitioned* into bundles. Yet, not all types of path drawings P allow the creation of a structure I. *Implicit* methods let paths (or path fragments) self-organize into bundles, by using variants of the mean shift operator well known in image processing [12]. This is done in a geometric setting via force-based methods [4,29,47,48,60] or by using faster, GPU-parallelized, image-based methods [32,41,50,66]. Implicit methods can treat any path-set P; yet, controlling which paths get bundled together is much harder.

Our proposal combines the advantages of explicit and implicit methods: We explicitly model parts of the dataset P by bundles, so we can trace back a bundle to the subset of P it represents; however, we do not require an explicit support structure I to be constructed from P, so we keep the advantages of implicit methods.

Q3: Data manipulation: By changing a bundled drawing interactively, several goals can be achieved, most notably supporting *select*, *navigate*, *filter* and *arrange* tasks [6,42]. Several types of changes can be applied to bundles: Relaxation can be used to (linearly) interpolate between raw and bundled curves. Introduced by Holten [28], virtually all bundling methods use this mechanism to control the bundling tightness; changing the interpolation interactively also allows one to visually link bundled to un-bundled elements and thus to see what a bundle contains. *B* can also be applied (or prevented) locally on P, based on the user's point of interest [34], with variants known as fisheye and bring & go techniques [38,64] and edge plucking [70]. Such techniques are refined in more recent papers [43,57]. However, all such techniques essentially change the bundled drawing and *not* the underlying data. We extend the above techniques to allow interactive manipulation of the bundled representation to change the large and/or complex dataset P. Similar techniques were used to manipulate *e.g.* shapes via control points, or spatial regions in 2D visualizations via morphing [7]. Based on this, we generate a new curve set P' whose bundling reflects the manipulation, and whose statistical properties (in terms of spatial spread and local shape) are close to the original curve set P.

3 Functional bundling framework

Our bundling proposal, based on functional statistics, is explained next. We first outline how we represent discretized curves as points in a function space (Sec. 3.2), thereby covering $Q1$. Next we explain how to represent a whole bundle by the concept of principal curves (Sec. 3.3), thereby covering $Q2$. Finally, we show how we manipulate data by manipulating the bundles' principal curves and resampling the distributions these capture (Sec. 3.4), thereby covering $Q3$. Section 3.5 gives the pseudocode for our full method, for reading convenience.

3.1 Curve clustering

As we shall see when we present our statistical bundle model (Sec. 3.3), we model a curve-set C by a mean curve and variance with respect to this mean. This works well if the variance of curves over C is not too large, just as PCA works well for point clouds that can be well approximated by hyper-ellipsoids. To enforce this, we pre-partition the input path-set P into a set of clusters C_i having limited intra-cluster variance. Implementation details are given in Sec. 3.5.

3.2 Curve functional modeling

Both our input and bundled curves are sets of $\mathbb{R}^d$ curves. In contrast to most existing bundling methods, we do not have any constraint on d, *i.e.* we can handle 2D, 3D, or higher-dimensional curves. Let us consider our first question $Q1$—how to model such curves γ. If we used standard multivariate statistics, γ would be a set of d-dimensional samples [44,53,69]. This can lead to a loss of prior knowledge about the data, such as control on curve shape or smoothness. Functional statistics [56] is a recent research area that aims to overcome the limitations of multivariate data representation when applied to functions and, in particular, curves. It models curves as *functions* that are points in an infinite-dimensional function space. We choose the space L^2 of all square integrable functions, as this space is commonly used when modeling curves [56]. For smooth curves (and bundled curves) such as ours, we also assume that the first two derivatives belong to L^2. Altogether, we will model our curves as functions in the Sobolev space

$$\mathbb{W}^2 = \left\{ f \in C^1([0,1],\mathbb{R}), f' \text{ abs. cont.}, \int_0^1 f(x)^2 + f''(x)^2 dx < +\infty \right\}. \tag{2}$$

We can now define the *norm* of a function $f \in \mathbb{W}^2$ as

$$\|f\|_{\mathbb{W}^2} = \left(\int_0^1 f(x)^2 + \beta f''(x)^2 dx \right)^{1/2}. \tag{3}$$

Here, the term $f(x)^2$ gives the distance of f to 0, thus capturing reconstruction accuracy; the term $f''(x)^2$ captures local curvature, which allows constraining the smoothness of our bundles. The parameter β is discussed below.

Discrete implementation: In bundling applications, curves come as discrete (sampled) objects. We model a curve as a set of sample points $\gamma = (x_1, \dots x_n)$, $x_i \in \mathbb{R}^d$, taken at parametric positions $(t_1, \dots, t_n)$, $t_i \in [0,1]$. From now on, we will use functional notations, *e.g.* $\gamma(t)$, with $t \in [0,1]$, to denote continuous functions in $\mathbb{W}^2$, and notations without parameters, *e.g.* γ, to denote their discrete (sampled) counterparts. To go to our functional setting, we first find the continuous curve that γ samples by seeking for a function $f \in \mathbb{W}^2$ with minimal norm and which minimizes the penalized sum-of-squares residual [56] given by

$$e = \sum_{i=1}^{n} (f(t_i) - x_i)^2 + \beta \int_0^1 f''(s)^2 ds. \tag{4}$$

Note that Eqn. 4 is derived from Eqn. 3 by stating that f passes close to the points of γ. In both equations, $\beta \geq 0$ balances between reconstruction error and smoothness: When β is small, approximating curves f go close to the points x_i; large β values yield smoother curves, but a less accurate approximation. The preset $\beta = 0.1$ was used for all figures in this paper. Using reproducing kernel theory [1], we can next express any f as a linear combination of basis functions

$$f(t) = \alpha_0 + \alpha_1 t + \sum_{i=2}^{n+2} \alpha_i K(t, t_i) \tag{5}$$

where $\alpha_0, \dots, \alpha_{n+2}$ are real-valued coefficients computed so as to minimize e. K is a cubic spline kernel, which is often used to approximate curves with continuous second-order derivatives [15], given by

$$K(t,s) = \frac{\max((t-s)^3, 0)}{6} - \frac{t^2 s}{2} - \frac{t^3}{6}. \tag{6}$$

Separately, note that any $f \in \mathbb{W}^2$ with $f(0) = f'(0)$ obeys the so-called reproducing property

$$f(t) = \int_0^1 \frac{\partial^2 K}{\partial s^2}(t,s) f''(s) ds. \tag{7}$$

Denoting $a = (\alpha_0, \dots, \alpha_{n+2})$ the vector of unknown coefficients that we need to find to minimize e, and using the expression of f as a sum of kernel functions (Eqn. 5) and the property in Eqn. 7, we can write the error e (Eqn. 4) in matrix-vector notation as

$$e = \|\gamma - Ga\|^2 + \beta a^T \tilde{G} a. \tag{8}$$

where G and $\tilde{G}$ are the following matrices

$$G = \begin{pmatrix} 1 & t_1 & K(t_1,t_1) & \dots & K(t_1,t_n) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & t_n & K(t_n,t_1) & \dots & K(t_n,t_n) \end{pmatrix} \tag{9}$$

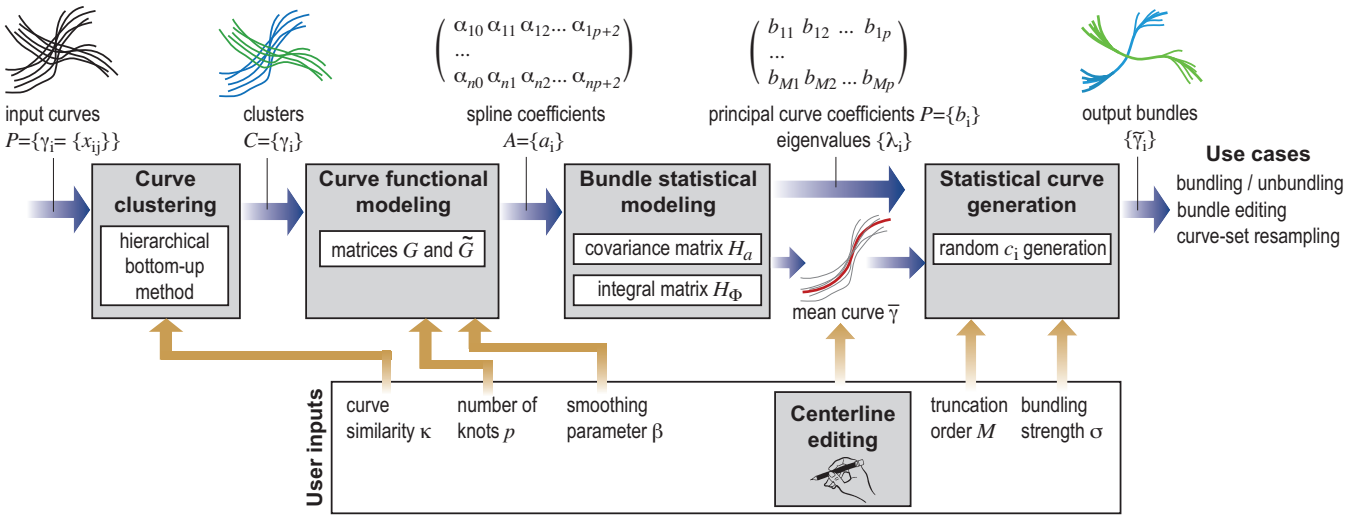

Fig. 1. Functional bundling steps: clustering (Sec. 3.1), curve modeling (Sec. 3.2), bundle modeling (Sec. 3.3), and curve generation (Sec. 3.4).

and

$$\tilde{G} = \begin{pmatrix} 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & K(t_1,t_1) & \dots & K(t_1,t_n) \\ \vdots & \vdots & \vdots & & \\ 0 & 0 & K(t_n,t_1) & \dots & K(t_n,t_n) \end{pmatrix}. \tag{10}$$

To minimize e, we equate its gradient from Eqn. 8 to zero and obtain

$$\left(G^T G + \beta \tilde{G}\right) a = G^T \gamma. \tag{11}$$

This is a linear system with positive-definite matrix and unknown vector a, easy and fast to solve numerically using LAPACK [40].

Discrete curves $\gamma \in P$ usually have different sample point counts n and sampling locations t_i. This implies a different expression of f (Eqn. 5) for every γ, which complicates our work. To solve this, we fix the spline basis used in computation, then find the best approximation of curves on it: Assuming spline kernels $K(t,\eta_i)$ are located at p so-called knot locations $(\eta_1,\dots,\eta_p)$ in the interval $[0,1]$, we obtain f by solving the same linear system Eqn. 11, but with the modified matrices

$$G = \begin{pmatrix} 1 & t_1 & K(t_1,\eta_1) & \dots & K(t_1,\eta_p) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & t_n & K(t_n,\eta_1) & \dots & K(t_n,\eta_p) \end{pmatrix} \tag{12}$$

and

$$\tilde{G} = \begin{pmatrix} 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & K(\eta_1,\eta_1) & \dots & K(\eta_1,\eta_p) \\ \vdots & \vdots & \vdots & & \\ 0 & 0 & K(\eta_p,\eta_1) & \dots & K(\eta_p,\eta_p) \end{pmatrix}. \tag{13}$$

The matrix $\tilde{G}$ has to be computed only once for a path-set P as it only depends on the fixed knot locations η_i. The speed of this fixed-knot evaluation is practically identical to that of the variable-knot version, as the only thing that changes is the content of the matrices G and $\tilde{G}$, and the number of knots p is of the same order of magnitude as the sample point count n.

Summary: Given a curve-set $P = \{\gamma_i\}$, we compute a matrix A whose rows a_i represent each a curve γ_i in terms of spline coefficients. (see also Fig. 1). We next use A to analyze and manipulate P.

3.3 Bundle statistical modeling

Consider now a group of curves $C = \{\gamma_1,\dots,\gamma_N\} \subset P$. We model C by its mean curve $\overline{\gamma}$ and variance around it, thereby answering Q2. We thus define *bundling* as the operation that reduces the deviation of curves in C from $\overline{\gamma}$. For this, we proceed as follows. For a curve set C, we first need to model its variability. A classical hypothesis is that C comes from an underlying hidden stochastic process Γ: $\Omega \times [0,1]$ where Ω is the probability space of all possible outcomes. We further assume that for any $t \in [0,1]$, $E_\omega[\|\Gamma(t,\omega)\|^2] < +\infty$. For any $t \in [0,1]$, the expectation $E_\omega[\Gamma(t,\omega)]$ is well-defined; we estimate it by the empirical mean curve estimator $\overline{\gamma}$ defined as

$$\overline{\gamma}(t) = \frac{1}{N}\sum_{i=1}^{N} \gamma_i(t). \tag{14}$$

The covariance function H for our curve-set C is defined for any (s,t) in $[0,1]^2$ as $E_\omega[\Gamma(s,\omega)\Gamma(t,\omega)^T]$. H is a $d \times d$ symmetric and positive matrix. Similarly to the mean, the empirical estimator for H is

$$\hat{H}(s,t) = \frac{1}{N}\sum_{i=1}^{N} \gamma_i(t)\gamma_i(s)^T. \tag{15}$$

Using $\hat{H}$, we can now explain the variability of C around its mean $\overline{\gamma}$ by using the well-known Karhunen-Loève expansion [62]:

$$\Gamma(t,\omega) = \overline{\gamma} + \sum_{i=1}^{+\infty} b_i(\omega)\phi_i(t) \tag{16}$$

where b_i are real-valued random variables and the functions ϕ_i obey

$$\int_0^1 \hat{H}(s,t)\phi_i(s)ds = \lambda_i \phi_i(t) \tag{17}$$

for some values $\lambda_i > 0$. In other words, ϕ_i are the (vector-valued) eigenfunctions of the covariance operator with eigenvalues λ_i. The random variables b_i are called the *scores* of the decomposition. They have zero mean and are pairwise uncorrelated. We call the curves $\phi_j, 1 \le j \le n$ the *principal curves* of the set C. Note that our principal curves are not related to the lexically-identical term introduced by Hastie *et al.* [27] and further refined by Erdogmus *et al.* [49]: There, given a *point cloud*, one computes a *single* principal curve that locally minimizes the sum of square distances to the points; in contrast, our *several* principal curves capture the variability of a set of curves in multiple orthogonal directions.

Discrete implementation: Let $x_{ij} = \gamma_i(t_{ij}), j = 1\dots n_i$ be the sample points of γ_i at positions $0 < t_{ij} < 1$. When all curves γ_i are sampled at the same positions, *i.e.* $t_{ij} = s_j, i = 1\dots N, j = 1\dots n$, we compute the mean curve via Eqn. 14 as $\overline{\gamma}(s_j) = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$. When sampling positions t_{ij} or sample counts n_i differ over C, we gather all samples x_{ij} and positions t_{ij} into a single set, then solve Eqn. 5 on this set to obtain $\overline{\gamma}$ (for full details, see Yao *et al.* [72]).

To find the principal curves ϕ_i, recall that all our functions $f \in \mathbb{W}^2$ are approximated using spline kernels (Sec. 3.2). Let $\Phi(t)$ be the vector of the p spline kernels introduced in Eqns. 12 and 13, *i.e.*

$$\Phi(t) = \begin{pmatrix} K(t,\eta_1) \\ \vdots \\ K(t,\eta_p) \end{pmatrix}. \tag{18}$$

We can thus write any function f with values in $\mathbb{R}^d$ as $f(t) = a^T\Phi(t)$, where a is a $p \times d$ matrix of real-valued coefficients. Each row of a contains one instance of the values $\alpha_0, \ldots, \alpha_{p+2}$ obtained by solving Eqn. 11 for one of the d dimensions. Let a_i and b_i be the matrices for the curve γ_i and principal curve ϕ_i, respectively. With these notations, the empirical covariance $\hat{H}$ from Eqn. 15 can be written as

$$\hat{H}(s,t) = \frac{1}{N}\sum_{i=1}^{N} \gamma_i(t)\gamma_i(s)^T = \Phi(t)^T H_a \Phi(s) \tag{19}$$

with

$$H_a = \frac{1}{N}\sum_{i=1}^{N} a_i a_i^T. \tag{20}$$

Next, by substituting $\hat{H}(s,t)$ from Eqn. 19 into Eqn. 17, we get

$$\Phi(t)^T H_a \left(\int_0^1 \Phi(s)\Phi(s)^T ds\right) b_i = \lambda_i \Phi(t)^T b_i. \tag{21}$$

Denoting $H_\Phi = \int_0^1 \Phi(s)\Phi(s)^T ds$, Eqn. 21 becomes

$$\Phi(t)^T H_a H_\Phi b_i = \lambda_i \Phi(t)^T b_i. \tag{22}$$

Since Eqn. 22 is valid for any $t \in [0,1]$, we get

$$\left(\int_0^1 \Phi(t)\Phi(t)^T dt\right) H_a H_\Phi b_i = \lambda_i \left(\int_0^1 \Phi(t)\Phi(t)^T dt\right) b_i \tag{23}$$

or, much more usefully, in matrix form

$$H_\Phi H_a H_\Phi b_i = \lambda_i H_\Phi b_i. \tag{24}$$

Note that we compute H_Φ only once as it depends only on the spline kernel knots η_i; and this computation involves only closed-form integrals. To speed up solving Eqn. 24, we compute the Cholesky decomposition $H_\Phi = L^T L$ with L lower-triangular and invertible, so Eqn. 24 becomes

$$L^T L H_a L^T L b_i = \lambda_i L^T L b_i. \tag{25}$$

Since L^T can be inverted, the above is equivalent to

$$(LH_aL^T)Lb_i = \lambda_i Lb_i \tag{26}$$

which we solve for Lb_i by a standard numerical eigensolver [24]. Next, we find b_i, the coefficients of principal curve ϕ_i, by simple back-substitution (since L is lower-triangular).

Summary: We capture the variation of a curve-set C by a mean curve $\bar{\gamma}$ and a (potentially infinite, see Eqn. 16) set of principal curves ϕ_i, described by their coefficient matrices b_i. We use these next to bundle and/or manipulate C, as follows.

3.4 Statistical curve generation

To perform both bundling and data manipulation (thereby answering Q3), we first fix a truncation order M for the infinite series in the Karhunen-Loève expansion (Eqn. 16). We can now generate a new curve $\tilde{\gamma}$ that is statistically similar to those in C by drawing coefficient vectors $c_i, i = 1 \ldots M$ from the respective laws of the random variables $b_i, i = 1 \ldots M$ and summing them, *i.e.* by computing

$$\tilde{\gamma}(t) = \bar{\gamma} + \sum_{i=1}^{M} c_i \phi_i(t). \tag{27}$$

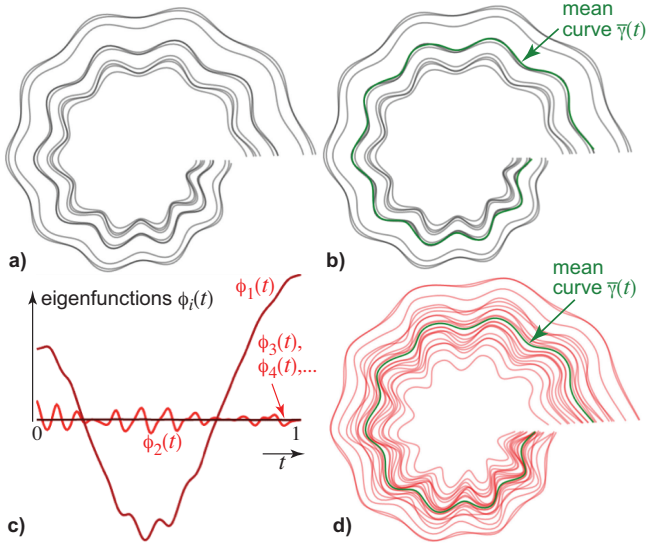

Fig. 2. Sample curve-set (a) and its reconstruction (b) using two eigenfunctions. Graph of eigenfunctions (c) and newly sampled curves (d).

For this, we need the distributions of b_i. We could estimate these from C using non-parametric kernel estimators [18]. If we assume that b_i are normally distributed, a simpler (and faster) way is to use the variance, which is given by the eigenvalues λ_i obtained when solving Eqn. 26. Generating a new curve $\tilde{\gamma}$ that fits the distribution in C is now easy:

1. Draw n random real numbers $c_1, \ldots, c_M$ according to independent, centered normal distributions with variances equal to the eigenvalues λ_i obtained from Eqn. 26;

2. Get the new curve $\tilde{\gamma}$ by using Eqn. 27.

This mechanism allows us to perform two types of operations:

Data manipulation: Replacing $\bar{\gamma}$ by any other curve $\gamma' \in \mathbb{W}^2$ in Eqn. 27 lets us create curves that approximate the distribution in C around any 'centerline', thereby manipulating the input curve-set P. If we use the original expansion coefficients of curves from C instead of random values c_i, we effectively create a new curve-set that has the same deviation from γ' as the original C has from $\bar{\gamma}$.

Bundling and unbundling: Varying the coefficients c_i from 0 to their observed values, *i.e.* replacing their values by σc_i in Eqn. 27, where $\sigma \in [0,1]$ is an interactively controlled parameter, smoothly relaxes the curves from $\bar{\gamma}$ to their initial positions, much like the relaxation introduced by Holten [28] (see Sec. 4.2 for examples). We dub this statistical curve generation *unbundling*: By analogy, if bundling creates a simplified representation of a path-set (in our case, mean and principal curves), then unbundling is the inverse process that takes this representation to generate a path-set. Note that this goes beyond the classical bundle relaxation used by most bundling techniques: We do not only control relaxation, via the values c_i, but can also generate *new* curves, and we can use a different bundle location (centerline) than the one given by the actual data.

For all tested datasets, setting $M \leq 5$ gives very accurate reconstructions. We show this for an artificial test set of undulating curves (Fig. 2a), which we reconstruct very well by only $M = 2$ eigenfunctions for the two largest eigenvalues (Fig. 2b). Indeed, we see that all ϕ_i for $i > 2$ are almost zero everywhere (Fig. 2c). Figure 2d shows how we can generate new curves (in red) which follow the shape of the initial curve-set within the constraints of the normal distribution assumption.

3.5 Implementation

We first cluster the input path-set P by bottom-up hierarchical agglomerative clustering [16] with a dissimilarity $\kappa(\gamma_1, \gamma_2)$ between two curves

Table 1. Datasets used for bundling in Fig. 3. For each dataset, we list its number of curves, computed clusters, sampling point count, total computation time, and other bundling papers which used it.

Dataset	curves	clusters	samples	time (secs.)	also used in
France	17275	232	50K	26	[19,32,50,66]
Paris	8620	91	23K	21	[50,66]
US airlines	2099	129	11K	5	[14,29,32,39]
US migrations	9780	174	30K	7	[19,29,32,66]
Radial	4021	46	30K	7	[19,32,63]

γ_1 and γ_2 equal to their Hausdorff distance, also used by earlier bundling techniques [19,63], defined as

$$d_H(\gamma_A,\gamma_B) = \sum_{x_i^A \in \gamma_A} d(x_i^A,\gamma_B) + \sum_{x_i^B \in \gamma_B} d(x_i^B,\gamma_A) \tag{28}$$

where

$$d(x \in \gamma_A,\gamma_B) = \|x - n_{\gamma_B}(x)\| \tag{29}$$

is the distance from a point $x \in \gamma_A$ to the closest point $n_{\gamma_B}(x) \in \gamma_B$ to x. We next define the dissimilarity between two curve-sets C_A and C_B as

$$\kappa(C_A,C_B) = \left(\frac{\sum_{\gamma_i \in C_A, \gamma_j \in C_B} \kappa(\gamma_i,\gamma_j)}{|C_A| \cdot |C_B|} \right) \cdot \left(\sum_{i=1}^{M} (\lambda_i^{C_A} - \lambda_i^{C_B})^2 \right) \tag{30}$$

where λ_i^C is the i^{th} largest eigenvalue of curve-set C. The variance-accounting second factor in Eqn. 30 makes comparing curve-sets more accurate than using only average-linkage distance (first factor), and much faster than full-linkage distances [36]. To efficiently compute closest points, we use a quadtree structure [2]. Having too few curves in a cluster makes the statistical curve generation (Sec. 3.4) unreliable, as these cannot truly create a distribution (that we next want to sample). Hence, if we find a cluster with less than 10 curves, and the user does not want to increase κ_{min}, we exclude these curves from the process—we simply render them unbundled. This makes sense, as we can then spot these 'outliers' which are dissimilar from all other bundled curves.

Algorithm 1 shows the computation of principal curves of a cluster C: We first derive the spline expansions of all curves $\gamma_i \in C$, see Sec. 3.2 (GETCURVEEXPANSION). Next, we compute the principal curves of C, see Sec. 3.3 (GETPRINCIPALCURVES). Finally, we create new curves around a (possibly modified) centerline, see Sec. 3.4 (Algorithm 2, GETNEWCURVE). In the above pseudocode, all point coordinates are scalars ($d = 1$) for writing simplicity. Extending to higher dimensions $d > 1$ is trivial, see Algorithm 3 (GETCURVEPOINT). Implementing Algorithms 1..3 is simple: We only need real-valued matrices and vectors (as data structures), a linear solver, and an eigensolver.

4 APPLICATIONS

We present next applications of our functional bundling for classical graph bundling (Sec. 4.1), trail data manipulation (Sec. 4.2), vector field visualization (Sec. 4.3), and tensor field visualization (Sec. 4.4).

4.1 Graph and trail-set bundling

We first use our method for classical graph or trail-set bundling: Given such a dataset P, we bundle it by computing the per-cluster functional decomposition of paths $\gamma_i \in P$, and next drawing these with coefficient values c_i close to zero. As explained in Sec. 3.4, this generates tight bundles. Figure 3 shows results for several well-known datasets in bundling literature: *France* is a one day of aircraft trails over French airspace; *Paris* selects flights that landed, took over, or overflew Paris in the *France* dataset; *US airlines* has flights between US airports over one year; *US migrations* captures population movements over the US over one year; and *Radial* is a function call graph in a software system. The *France* and *Paris* datasets contain 3D trails (longitude, latitude, altitude). To our knowledge, only 2D versions (latitude, longitude) of these datasets have been bundled so far [19,32,50,66]. We directly achieve 3D bundling by our functional method, see Fig. 4. All other datasets contain 2D trails. Table 1 shows details, including computation times. For more insight, Fig. 3 (right column) shows the computed

Algorithm 1 Curve modeling and principal curves computation

procedure GETCURVEEXPANSION(A,C,s,t) ▷ A: returned coefficients matrix; C: curve sample-points (row-wise); s: sample positions vector; t: kernel locations vector
  $n \leftarrow$ **length**(s); $m \leftarrow$ **length**(t)
  for $i = 1,n$ **do** ▷ Compute matrix G (Eqn. 12)
    $G[i,1] \leftarrow 1$; $G[i,2] \leftarrow s[i]$
    for $j = 1,m$ **do**
      $G[i,j+2] \leftarrow K(s[i],t[j])$
    end for
  end for
  for $i = 1,m$ **do** ▷ Compute matrix $\tilde{G}$ (Eqn. 13)
    for $j = 1,m$ **do**
      $\tilde{G}[i,j] \leftarrow K(t[i],t[j])$
    end for
  end for
  $M \leftarrow G^T G + \beta \tilde{G}$
  $A \leftarrow M^{-1} G^T C$ ▷ Inverse M^{-1} is not computed; a linear system is instead numerically solved (Eqn. 11)
end procedure
procedure GETHPHIMATRIX(L,t) ▷ L: returned Cholesky factorization of the H_Φ matrix; t: kernel locations vector
  $n \leftarrow$ **length**(t)
  for $i = 1,n$ **do**
    $L[i,j] \leftarrow \int_0^1 K(t_i,s) * K(t_j,s) ds$ ▷ Integral is closed form
  end for
  Cholesky(L)
end procedure
procedure GETCOEFFSCOVARIANCE(H_a,A) ▷ H_a: returned covariance matrix; A: curve coefficient vectors (row-wise)
  $(n,m) \leftarrow$ **size**(A) ▷ (n,m) = (#rows,# columns) of A
  $H_a \leftarrow n^{-1} A^T A$ ▷ Covariance computed using Eqn. 20
end procedure
procedure GETPRINCIPALCURVES(P,v,A,t) ▷ P: returned matrix of principal curves coefficients (row-wise); v: returned eigenvalues vector; A: curve coefficients matrix (row-wise); t: kernel locations
  $n \leftarrow$ **length**(t)
  GetCoeffsCovariance(H_a,A)
  GetHPhiMatrix(L,t)
  $M \leftarrow L H_a L^T$
  eigen(M,P,v) ▷ Compute numerically eigenvectors of M (in P) and eigenvalues (in v), see Eqn. 26
  $P \leftarrow L^{-1} P$
end procedure

Algorithm 2 Statistical curve generation

1: **procedure** GETNEWCURVE(a,b,P,v) ▷ P,v are same as in **GetPrincipalCurves**; b: coefficient vector for $\overline{\gamma}$; a: returned vector of coefficients for the randomly-generated curve
2: $n \leftarrow$ **length**(b)
3: **for** i=1,n **do**
4: $a[i] \leftarrow b[i] + \sqrt{v[i]} \cdot P[i,:] \cdot$ **rnd**(n) ▷ **rnd**(n) draws a vector of length n from a normal distribution
5: **end for**
6: **end procedure**

Algorithm 3 Curve evaluation (sampling)

1: **procedure** GETCURVEPOINT(x,t,a,s)) ▷ x: returned point; t: parametric coordinate of x; a: coefficients vector for the curve that x is on; s: spline kernel locations
2: $n \leftarrow$ **length**(s)
3: **for** i=1,n **do**
4: $v[i] \leftarrow K(t,s[i])$ ▷ K is the spline kernel in Eqn. 6
5: **end for**
6: $x \leftarrow v^T a$
7: **end procedure**

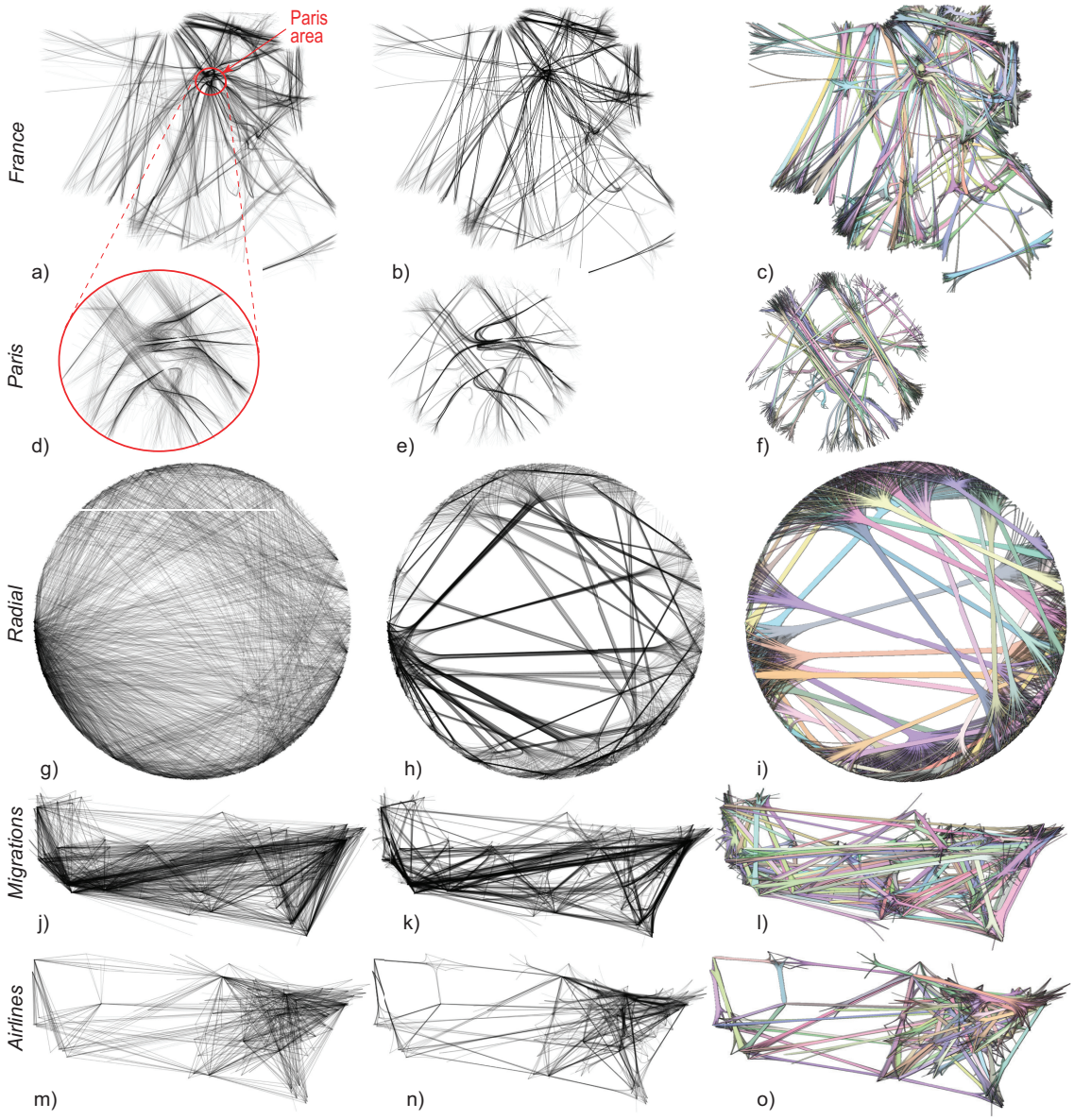

Fig. 3. Bundling by functional decomposition for graphs and trail-sets. Per column, left to right: raw curves, bundles, and colored clusters.

curve clusters, colored and shaded using the IBEB technique [63]. To obtain 'thick' cluster shapes, we use a relaxation $\sigma = 0.2$.

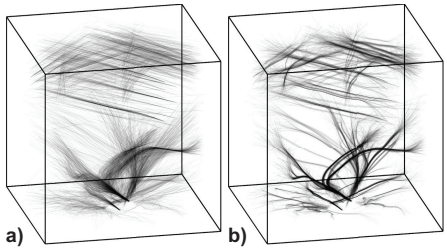

Fig. 4. Bundling a 3D trail-set. (a) raw data; (b) bundles.

As visible, our functional method produces similar results to those yielded by earlier methods [29, 32, 63, 66]. Of course, differences exist, *e.g.* in terms of bending, thickness, and number of bundles produced; and one can argue that other techniques produce more insightful and/or more aesthetically pleasing results than ours. We did not explore the parameter space of our method in detail to see how it can be tuned to imitate (or improve) upon existing methods, which is surely to be considered next. As Tab. 1 shows, our current implementation is much slower than state-of-the-art GPU-based methods [41, 66]. However, as already explained, our method can directly work in more than two dimensions, whereas all existing fast bundling methods known so far are designed (and optimized) for 2D datasets. Apart from 3D bundling, our contributions are in the novelty of functional bundling, the representation of bundles as explicit objects (to our knowledge, only done so far by SBEB [19]), and our ability to *manipulate* the underlying data via bundling, the latter being demonstrated next.

4.2 Trail-set manipulation for ATC planning

We next show how functional bundling enables users to easily manipulate a path-set via its simplified (bundled) representation. Our data is a set of recorded aircraft trails in 3D space (longitude, latitude, height). In air-traffic control (ATC), flight routes consist of an ordered sequence of spatio-temporal references called *beacons*. Aircraft usually follow flight routes but can deviate from them, within bounds, for safety or traffic optimization reasons. Routes are very complex to define in practice and have a huge impact on flight time, fuel consumption, and traffic complexity [71]. Also, routes are incrementally changed over time by ATC planners to optimize traffic flow (managing more aircraft) with

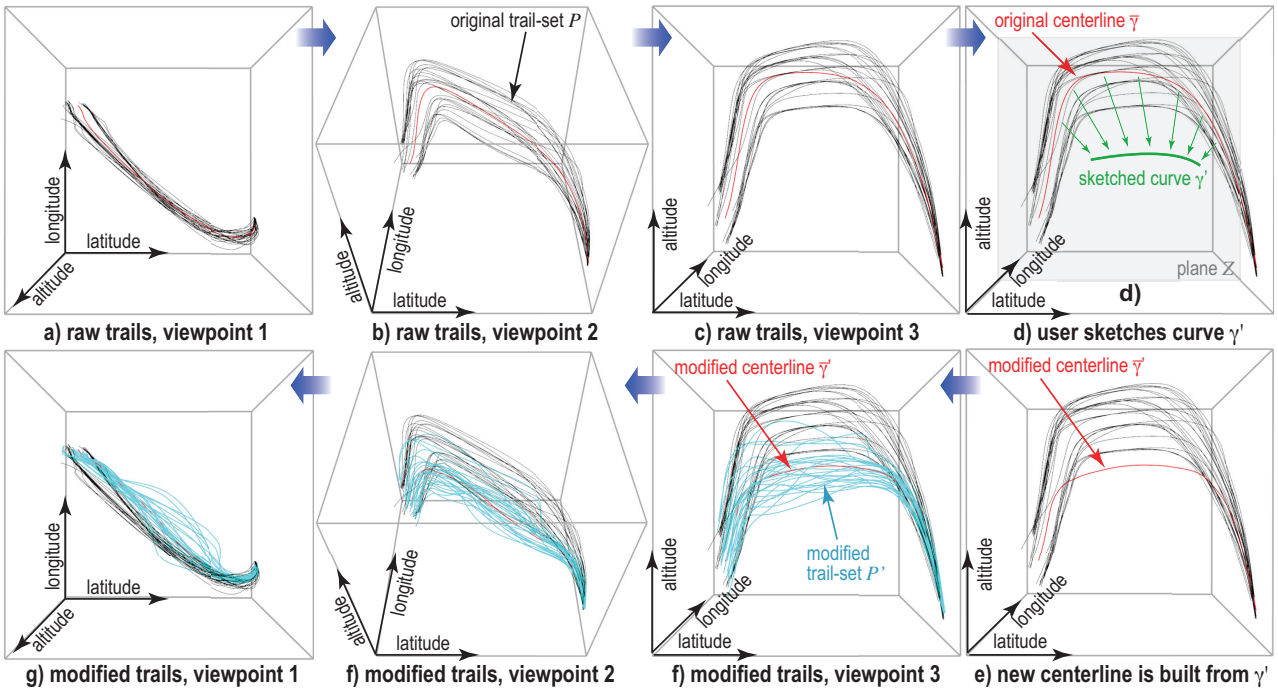

Fig. 5. Changing flight routes of a 3D flight dataset using functional bundling and user-edited flight routes. See Sec. 4.2.

higher safety and improved traffic fluidity, but also when new flights are scheduled. Each route change can have a large impact on a high (national or international) planning level, and change impact is very complex to forecast. Summarizing, the process of changing a route, based on existing and new flight data, and assessing the change impact, is expensive and laborious. We aim to improve this.

Functional bundling offers a new way to easily change flight routes based on actual traffic data and to assess change impact, as follows. Figures 5a-c show flight paths for a specific section of a flight route between Nice (southeast of France) and Paris (north of France) from three viewpoints. Flights first ascend, then descend—see view (c). A planner wants to improve average aircraft fuel consumption (for all flights on this route) by changing the route to reduce the ascent and descent parts to lower altitudes. For this, the planner interactively modifies the mean path $\overline{\gamma}$ of the trail-set P (shown red in Fig. 5). We support this by a simple 3D curve-editing tool that allows sketching a target curve γ' (Fig. 5d, green). The curve is sketched in the view plane; its depth coordinates are set to the average depths of the trail-set P as seen from the current viewpoint—γ' thus lives in the half-transparent plane Z (Fig. 5d). After sketching, the mean curve $\overline{\gamma}$ is smoothly attracted towards γ' (Fig. 5d). A To incrementally model a reasonably complex 3D centerline $\overline{\gamma}'$ with limited effort, a few sketch steps, done from different viewpoints, are typically enough (Fig. 5e). Next, our method changes the input trail-set P so as to follow $\overline{\gamma}'$ *and* to respect their original spatial spread in P (Sec. 3.4). The new trails P' (cyan in Figs. 5f-h) can be compared from the original viewpoints with the input trails (shown black). Using P', we can now compute the fuel savings, and repeat the loop if we want to further improve. This scenario answers the question "how much fuel can we save if all *existing* flights Nice-Paris flew lower but having the same spatial distribution?". We can also address a more involved question "how much fuel can we save if *extra* flights Nice-Paris were to appear, following the same spatial distribution as the current ones, along the new low-altitude route?". For this, we use the curve-generation method in Sec. 3.4 to create new trails that follow our desired route γ' and also follow the spatial distribution in P.

The entire above analytic loop was executed by several specialist ATC planners from the French civil aviation, with over 10 years of experience in the field, who also came up with the fuel-saving question. The loop took a few minutes, which the planners stated to be much more cost-effective for exploring 'what if' ATC-planning scenarios than their current tools which require manually changing *every single* flight-path when a route changes by changing the path's tens of control points. Our current 3D sketching tool is, of course, basic; more advanced ones can be imagined. One could also argue that editing a 3D curve-set by changing its centerline could be done by simpler geometry-based techniques. However, generalizing this to $d > 3$ dimensions *and* ensuring that the edited curve-set has similar statistical properties (spatial spread, mean curve, and local curvature) as the original one, is far from trivial. Functional bundling offers these features by design. We do not know of any curve-set editing techniques that guarantee such properties; for instance, classical spline or NURBS editing via control points can easily create kinks, loops, and fold-backs in a curve. Separately, we see how functional bundling can easily handle 3D trail-sets; more dimensions, *e.g.* data on flight speed, fuel live consumption, or wind force, can be immediately added, as we handle any number of dimensions (Sec. 3.5). Finally, we note the similarity of our use-case with transmogrifiers [7]: Both techniques modify a visualization by sketching simple shapes directly in screen space. However, our method modifies not just the visualization, but the underlying *data*, with statistical guarantees offered.

4.3 Vector field visualization

Simplified visualizations of vector fields have used streamline clusters [37, 45, 73] depicted by representative streamlines from each cluster, for tasks like 3D clutter and occlusion reduction and spotting salient patterns. We achieve similar results: We start with a large set of 3D streamlines γ_i created by densely seeding the flow domain, and bundle it by our method. Figure 6(a-e) show results for increasing simplification levels, *i.e.*, decreasing numbers of clusters, for a flow in a basin computed on a grid of $128 \times 85 \times 42$ cells [26], and a set of 4595 streamlines (183K sample points). For comparison, Fig. 6f shows two simplification levels for the same field computed with the algebraic-multigrid (AMG) method of Griebel *et al.* [26]. Our method captures well the salient flow structures, even at coarse levels, producing results in line with AMG. The inlet-outlet flow and the vortex-like region in Fig. 6a are visible even in the coarsest view (Fig. 6e). Next, in Fig. 6e we generate extra curves using the data distribution around the cluster centerlines (Sec. 3.4). We see how these curves obey the local flow structure and add detail without increasing clutter or occlusion. The image corresponding to Fig. 6e, without generated curves, is not shown for space reasons, but is easy to imagine—it contains curves centered in

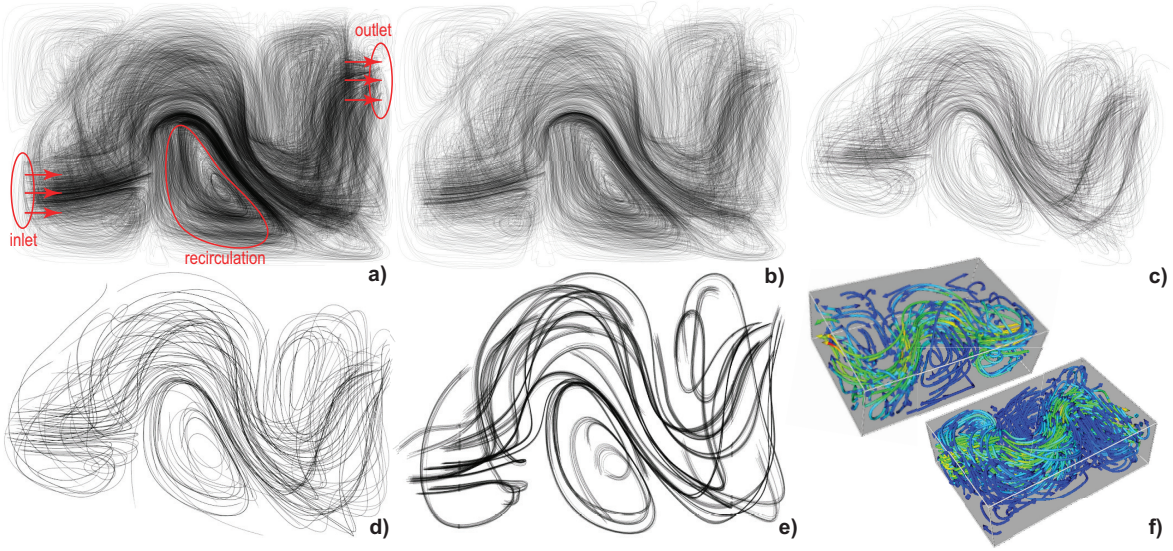

Fig. 6. (a-e) Progressive bundling of 3D streamlines for a flow field. (f) Comparison with multiscale flow simplification [26]. See Sec. 4.3.

the thick streamline 'bands' visible in Fig. 6e. Highly simplified images like Fig. 6e cannot be (easily) obtained by using fewer randomly-seeded streamlines, as this does not guarantee good coverage, but require more involved streamline seeding methods [45, 73]. The only other streamline bundling method we are aware of is hierarchical streamline bundles (HSB) [73]. Similarities, but also differences, exist with our work: (a) HSB uses carefully-placed seeds based on the field's salience; we do not assume any specific seeding or streamline spatial distribution. (b) Both methods use the same type of bottom-up hierarchical clustering and similar Hausdorff distances to compare paths. (c) both methods describe streamline clusters statistically; HSB defines a centerline as an explicit streamline being closest to all streamlines in a cluster, and measures the standard deviation from it; we propose the more detailed principal curves, which do not force the mean to be an existing input curve, and model variance by M eigencurves rather than simple standard deviation. (d) HSB defines *bundles* as the streamlines closest to a cluster's boundary, thereby aiming to capture cluster *boundaries*; we define them as a sampling of a cluster that minimizes variance, thereby aiming to capture the *inside* of clusters. Finally, comparing Tab. 1 with Tab. 1 in HSB [73] shows that we are one order of magnitude faster on similar datasets (number of cells, streamlines, sample points), even though HSB has a CPU-GPU implementation. Summarizing, we do not claim better simplified vector field visualizations than comparable methods—but we show that functional bundling is a serious contender.

4.4 Tensor field visualization

Finally, we consider the simplified visualization of fiber tracts obtained as streamlines of the major eigenvector of a diffusion tensor imaging (DTI) field. Such datasets have a spatially complex structure which makes them hard to explore [3]. To assist with this, simplified visualizations have been proposed. Early methods cluster the fibers and display a small set of representatives for each cluster [46,65], or cluster outlines [11]. Bottger *et al.* [4] extend the mean-shift bundling [32] to 3D connections between brain areas; since the input data is a graph rather than a trail-set, the deformations produced by bundling are not critical. Everts *et al.* propose a variant of the above which, when used with careful seeding, filtering, and smoothing of the tracts, reduces spatial clutter [21]. Given all above, it is arguably interesting to explore new ways of producing simplified visualizations of DTI fiber sets.

Figure 7 shows how functional bundling produces simplified fiber-set visualizations. We traced 150352 fibers from high fractional anisotropy areas in a $128 \times 128 \times 51$ DTI volume (dataset from Everts *et al.* [21]). We filtered fibers shorter than 2mm, yielding a set P of 120593 fibers (6.4M sample points), shown in Fig. 7a color-coded on direction, with alpha blending. This image has visible occlusion and clutter. Figures 7b,c show our functional bundling results for two clustering

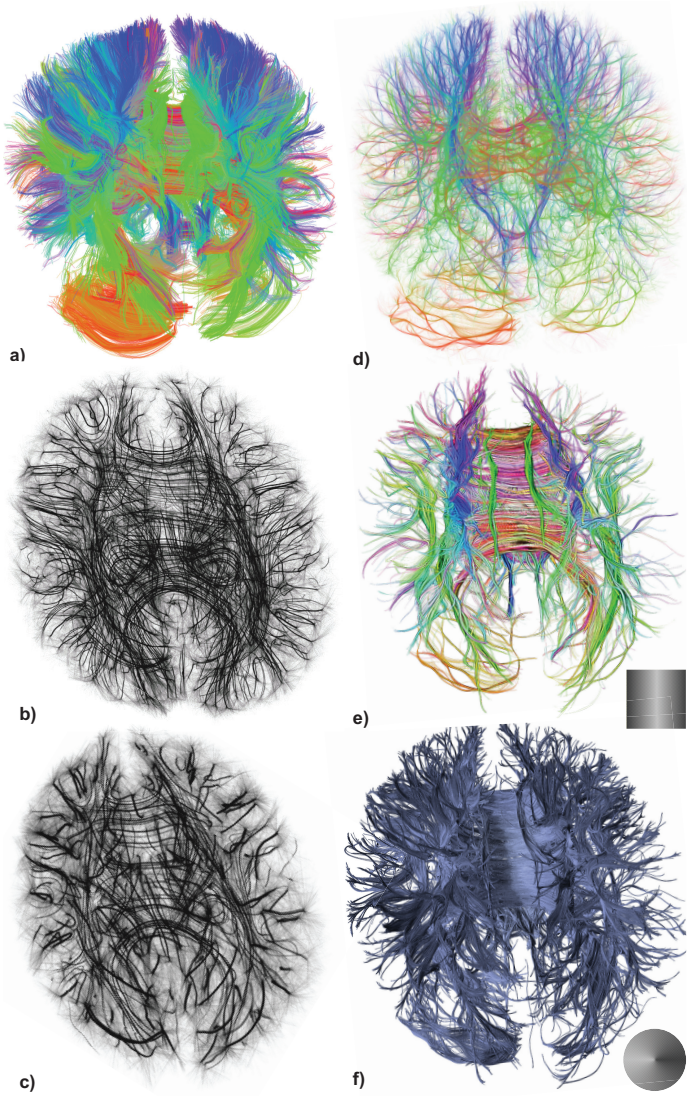

Fig. 7. DTI fiber tract bundling. (a) Input fibers. (d) Isotropic KDEEB bundling. Our method, with geometry-only similarity (b,c) and geometry-and-anisotropy similarity (e,f). Pseudo-shading effects are added using tube (e) respectively cone (f) billboard sprite textures. See Sec. 4.4.

levels. We draw only the cluster centerlines for clarity. The main connection patterns start becoming visible. Figure 7d shows the results of mean shift bundling [4], implemented by extending KDEEB [32] to 3D, which yields more wiggling bundles than our method. This is easily explained: KDEEB constrains bundle curvature only by *post-processing* the bundled curves by 1D Laplacian filtering; in contrast, we use curvature *explicitly* during bundling (parameter β, Eqn. 4). We can do more: Images (b-d), and all other fiber-bundling methods we know, use only the local relative position and proximity of the fibers. This creates gaps in *surfaces* formed by fibers, such as in the *corpus callosum* area in Figs. 7b-d. We fix this by a dissimilarity κ that uses the original DTI data: Given two fibers $\gamma_A \in P$, $\gamma_B \in P$, we define

$$\kappa(\gamma_A,\gamma_B) = \sum_{x_i^A \in \gamma_A} d_{DTI}(x_i^A,\gamma_B) + \sum_{x_i^B \in \gamma_B} d_{DTI}(x_i^B,\gamma_A) \qquad (31)$$

with

$$d_{DTI}(x \in \gamma_A,\gamma_B) = d(x,\gamma_B)\left(c_P(x)\left(\mathbf{e}_2(x)\cdot\frac{x-n_{\gamma_B}(x)}{\|x-n_{\gamma_B}(x)\|}\right)^k + 1\right) \qquad (32)$$

where d is given by Eqn. 29; $c_P(x) = \frac{2(\lambda_2-\lambda_3)}{\lambda_1+\lambda_2+\lambda_3}$, where $\lambda_1 \geq \lambda_2 \geq \lambda_3$ are the tensor eigenvalues, is the planar anisotropy [67] at x; $\mathbf{e}_2$ is the eigenvector for λ_2; and $k=2$ is a constant. Simply put, Eqn. 31 says that fibers in high planar-anisotropy regions are more similar to other fibers *outside* the plane than to neighbor fibers *in* the plane; elsewhere, κ is the Hausdorff distance of two 3D curves (compare Eqns. 31 and 28). This creates clusters that are orthogonal on fiber planes in regions of high c_P, so showing such clusters strongly bundled effectively 'pulls' fibers close to such planes into the planes themselves, *i.e.*, towards the location of the clusters' centerlines; in low c_P regions, we get isotropic bundling [4, 21]. Figures 7e,f show the resulting simplifications, rendered with view-plane-aligned billboard point sprites textured to imitate shaded-tube profiles and cone profiles (see insets in Figs. 7e,f respectively). Fiber sheets are now better visible, and have fewer gaps, than in Figs. 7b-d. The sprites create a pseudo 3D-like shading effect for the fiber-set (Fig. 7f). This is similar to depth-dependent halos (DDH) [22], but with important differences: Close fibers visually merge into sheets and tubes, much as in DDH; these structures are shaded in our case, whereas DDH renders only black shapes with white outlines. To conclude, we argue that, albeit not proven to surpass all comparable methods, our functional bundling yields visually compelling low-clutter simplified visualizations of complex DTI fiber-sets.

5 Discussion

We discuss next several relevant aspects of our method.

Shape control: Our bundles are guided by the mean curves computed by functional decomposition from the clustered path-set. Our clusters (Sec. 3.1) are coherent, so bundles will follow the input data with *small* deviations only. This contrasts to most, if not all, implicit bundling methods which strongly deform the input curves [4, 19, 29, 32, 50, 66]. The deformation amount in a bundle, given by the eigenvalues λ_i (Sec. 3.3), could be used to quantify, visualize, or limit the bundling extent, in contexts where path positions encode relevant data.

Multidimensionality: While we only show results for $d = 3$ dimensions (aircraft trails, streamlines, and DTI fibers), our entire method, including its implementation, can directly handle higher dimensions (Sec. 3). Except FFTEB [41], this is the only bundling method we know of that generalizes to $d > 3$.

Data manipulation: Functional decomposition allows modeling, but also manipulation of, a path-set. Besides the ATC planning use-case (Sec. 4.2), we could *e.g.* use exactly the same simple and fast procedure to re-organize a bundled drawing, *e.g.* to prevent distortions in some areas or to emphasize patterns of interest [32].

Parameters: We use the following parameters for all figures in this paper (see also Fig. 1): The dissimilarity κ tells which curves end in the same cluster. Good κ presets for isotropic bundling are given in earlier bundling papers [19, 46, 73]; for DTI fiber bundling, we propose a custom κ (Eqn. 31). The number of knots of our spline basis (Sec. 3.2) is preset to $p = 50$. To ensure a good approximation between discrete curves and the spline-based representations fitted to them, we use a relatively high number of curve sampling points (n between 50 and 100). This also makes n of the same magnitude order as p. The smoothing β (Eqn. 4) is preset to 0.2. Higher values yield smoother, but less well approximating, bundles. The truncation order is preset to $M = 5$ (Sec. 3.4). Finally, the bundling strength σ is interactively set to values in $[0,1]$ to navigate between the original and fully bundled curves (Sec. 3.4).

Table 2. 3D datasets used for functional bundling in Secs. 4.2-4.4.

Dataset	curves	clusters	samples	time (secs.)
ATC flights (Sec. 4.2)	4438	132	117K	6
Streamlines (Sec. 4.3)	4595	241	183K	46
DTI fibers (Sec. 4.4)	120593	608	6.4M	143

Limitations: If paths in a cluster differ too much, the computed centerlines $\overline{\gamma}_i$ contain spurious curls and twists, so the statistical generation (Sec. 3.4) can create curves that may not resemble the input ones. This is analogous to using PCA to characterize a point distribution which deviates too much from a hyper-ellipsoid. A related issue that we assume the coefficients b_i to be normally distributed. This generates random curves which are normally distributed around $\overline{\gamma}_i$ (Sec. 3.4). This normal distribution may differ from the distribution of the actual input curves. Studying how to better capture this actual distribution in the generation process is left for future work.

Our CPU-based functional decomposition is linear in cluster size. Table 2 lists the statistics for the 3D datasets used in Secs. 4.2-4.4. About 80% of our entire pipeline cost (Tabs. 1 and 2) is the bottom-up path clustering [16], which is $O(nN^2 \log N)$ for N paths each of n sample points on average. Hence, functional bundling by itself is reasonably fast and could benefit from faster clustering techniques [17, 20, 61]. Testing such algorithms, *e.g.* following recommendations for trail clustering [46], and using GPU-based linear-algebra tools to implement Algorithm 1, can make our method competitive with the fastest bundling techniques [32, 41, 66].

6 Conclusion

We have presented a new method for visual simplification of graphs and trail-sets based on functional decomposition. We model curves as linear combinations of piecewise polynomial (spline) basis functions with associated expansion coefficients, and we model path sets as a mean function with its associated expansion plus a set of principal curves expressed via the same basis functions. Trails correspond one-to-one to expansion coefficients which allows a simple generation of arbitrary curves that are statistically similar to the given ones. We use this technique in two ways: First, we create bundled views of the data based on the centroid functions. Secondly, we allow deforming the centroid function to manipulate the underlying data in a simple way while preserving statistical properties. Our method handles path-sets in any dimension, has a sound mathematical foundation, and can be implemented using standard linear algebra tools. We demonstrate our proposal for the visual simplification of graph drawings, 3D aircraft trail-sets, 3D streamlines and DTI fiber tracts, and the interactive construction and analysis of air-traffic flight routes.

Many extensions are readily possible. Principal curves allow a direct way to quantify, visualize, and limit the amount of deformation that bundling creates, which is useful when one wants to analyze and control such deformations. Secondly, functional bundling can simplify path-sets of more than three dimensions. This opens interesting ways for the (visual) exploration of high-dimensional static and dynamic path-sets.

Acknowledgments

The authors acknowledge the support of the French National Agency for Research (Agence Nationale de la Recherche ANR) under the grant ANR-14-CE24-0006-01 project "TERANOVA" and the SESAR Research and Innovation Action Horizon 2020 under project "MOTO" (The embodied reMOte TOwer).

REFERENCES

[1] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, (68):337–404, 1950. doi: 10.1090/S0002-9947-1950-0051437-7

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998. Code available at `https://www.cs.umd.edu/~mount/ANN`. doi: 10.1145/293347.293348

[3] Y. Assaf and O. Pasternak. Diffusion tensor imaging (dti)-based white matter mapping in brain research: A review. *Journal of Molecular Neuroscience*, 34(1):51–61, 2008. doi: 10.1007/s12031-007-0029-0

[4] J. Böttger, A. Schäfer, G. Lohmann, A. Villringer, and D. Margulies. Three-dimensional mean-shift edge bundling for the visualization of functional connectivity in the brain. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):471–480, 2014. doi: 10.1109/TVCG.2013.114

[5] U. Brandes and D. Wagner. Using graph layout to visualize train interconnection data. *Journal of Graph Algorithms and Applications*, 4(3):135–155, 2000. doi: 10.1007/3-540-37623-2_4

[6] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2376–2385, 2013. doi: 10.1109/TVCG.2013.124

[7] J. Brosz, M. A. Nacenta, R. Pusch, S. Carpendale, and C. Hurter. Transmogrification: Causal manipulation of visualizations. In *Proc. of the 26th ACM Symposium on User Interface Software and Technology (UIST)*, pp. 97–106, 2013. doi: 10.1145/2501988.2502046

[8] K. Buchin, B. Speckmann, and K. Verbeek. Flow map layout via spiral trees. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2536–2544, 2011. doi: 10.1109/TVCG.2011.202

[9] K. Buchin, B. Speckmann, and K. Verbeek. Angle-restricted Steiner arborescences for flow map layout. *Algorithmica*, 72(2):656–685, 2015. doi: 10.1007/s00453-013-9867-z

[10] P. Caserta, O. Zendra, and D. Bodénès. 3D hierarchical edge bundles to visualize relations in a software city metaphor. In *Proc. 6th International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pp. 1–8, 2011. doi: 10.1109/VISSOF.2011.6069451

[11] W. Chen, S. Zhang, S. Correia, and D. S. Ebert. Abstractive representation and exploration of hierarchically clustered diffusion tensor fiber tracts. *Computer Graphics Forum*, 27(3):1071–1078, 2008. doi: 10.1111/j.1467-8659.2008.01244.x

[12] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. doi: 10.1109/34.1000236

[13] B. Cornelissen, A. Zaidman, D. Holten, L. Moonen, A. van Deursen, and J. J. van Wijk. Execution trace analysis through massive sequence and circular bundle views. *Journal of Systems and Software*, 81(12):2252–2268, 2008. doi: 10.1016/j.jss.2008.02.068

[14] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1277–1284, 2008. doi: 10.1109/TVCG.2008.135

[15] C. De Boor. *A Practical Guide to Splines*. Number 27 in Applied Mathematical Sciences. Springer, 1978. doi: 10.1137/1022106

[16] M. de Hoon, S. Imoto, J. Nolan, and S. Myiano. Open source clustering software. *Bioinformatics*, 20(9):14–53, 2004. Code available at `bonsai.hgc.jp/~mdehoon/software/cluster`. doi: 10.1093/bioinformatics/bth078

[17] D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364, 1977. doi: 10.1093/comjnl/20.4.364

[18] A. Delaigle and P. Hall. Defining probability density for a distribution of random functions. *Annals of Statistics*, 38(2):1171–1193, 2010. doi: 10.1214/09-AOS741

[19] O. Ersoy, C. Hurter, F. Paulovich, G. Cantareiro, and A. Telea. Skeleton-based edge bundling for graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2364–2373, 2011. doi: 10.1109/TVCG.2011.233

[20] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 226–231. AAAI Press, 1996. doi: 10.1.1.121.9220

[21] M. H. Everts, E. Begue, H. Bekker, J. B. T. M. Roerdink, and T. Isenberg. Exploration of the brain's white matter structure through visual abstraction and multi-scale local fiber tract contraction. *IEEE Transactions on Visualization and Computer Graphics*, 21(7):808–821, 2015. doi: 10.1109/TVCG.2015.2403323

[22] M. H. Everts, H. Bekker, J. B. T. M. Roerdink, and T. Isenberg. Depth-Dependent Halos: Illustrative Rendering of Dense Line Data. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1299–1306, 2009. doi: 10.1109/TVCG.2009.138

[23] M. Giereth, H. Bosch, and T. Ertl. A 3D treemap approach for analyzing the classificatory distribution in patent portfolios. In *Proc. Visual Analytics Science and Technology (VAST)*, pp. 189–190, Oct 2008. doi: 10.1109/VAST.2008.4677380

[24] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996. doi: doi.org/10.1137/1028073

[25] M. Graham and J. Kennedy. Using curves to enhance parallel coordinate visualisations. In *Proc. of the 7th International Conference on Information Visualisation*, pp. 10–16, 2003. doi: 10.1109/IV.2003.1217950

[26] M. Griebel, T. Preusser, M. Rumpf, M. A. Schweitzer, and A. Telea. Flow field clustering via algebraic multigrid. In *Proc. IEEE Visualization*, pp. 35–42, Oct 2004. doi: 10.1109/VISUAL.2004.32

[27] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989. doi: 10.1080/01621459.1989.10478797

[28] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006. doi: 10.1109/TVCG.2006.147

[29] D. Holten and J. J. van Wijk. Force-directed edge bundling for graph visualization. *Computer Graphics Forum*, 28(3):670–677, 2009. doi: 10.1111/j.1467-8659.2009.01450.x

[30] C. Hurter. *Image-Based Visualization: Interactive Multidimensional Data Exploration*. Synthesis Lectures on Visualization. Morgan & Claypool, 2015. doi: 10.2200/S00688ED1V01Y201512VIS006

[31] C. Hurter, O. Ersoy, S. I. Fabrikant, T. R. Klein, and A. C. Telea. Bundled visualization of dynamicgraph and trail data. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1141–1157, 2014. doi: 10.1109/TVCG.2013.246

[32] C. Hurter, O. Ersoy, and A. Telea. Graph bundling by kernel density estimation. *Comput. Graph. Forum*, 31(3):865–874, 2012. doi: 10.1111/j.1467-8659.2012.03079.x

[33] C. Hurter, O. Ersoy, and A. Telea. Smooth bundling of large streaming and sequence graphs. In *2013 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 41–48, 2013. doi: 10.1109/PacificVis.2013.6596126

[34] C. Hurter, A. Telea, and O. Ersoy. MoleView: An Attribute and Structure-Based Semantic Lens for Large Element-Based Plots. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2600–2609, 2011. doi: 10.1109/TVCG.2011.223

[35] J. Jacques and C. Preda. Functional data clustering: A survey. *Advances in Data Analysis and Classification*, 8(3):231–255, 2014. doi: 10.1007/s11634-013-0158-y

[36] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, Sept. 1999. doi: 10.1145/331499.331504

[37] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Proc. of the Eurographics Workshop on Visualization in Scientific Computing (ViSC)*, pp. 43–55. Springer, 1997. doi: 10.1007/978-3-7091-6876-9_5

[38] A. Lambert, D. Auber, and G. Melancon. Living flows: Enhanced exploration of edge-bundled graphs based on GPU-intensive edge rendering. In *Proc. 14th International Conference on Information Visualisation*, pp. 523–530, 2010. doi: 10.1109/IV.2010.78

[39] A. Lambert, R. Bourqui, and D. Auber. Winding roads: Routing edges into bundles. *Computer Graphics Forum*, 29(3):853–862, 2010.

[40] LAPACK. Lapack linear algebra library. `http://www.netlib.org/lapack`, 2017.

[41] A. Lhuillier, C. Hurter, and A. Telea. FFTEB: Edge bundling of huge graphs by the Fast Fourier Transform. In *Proc. IEEE PacificVis*, 2017.

[42] A. Lhuillier, C. Hurter, and A. Telea. State of the art in edge and trail bundling techniques. *Computer Graphics Forum*, 2017. doi: 10.1111/cgf.13213

[43] S.-J. Luo, C.-L. Liu, B.-Y. Chen, and K.-L. Ma. Ambiguity-free edge-bundling for interactive graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(5):810–821, 2012. doi: 10.1109/TVCG.2011.104

[44] H. Mark and J. Workman. *Statistics in Spectroscopy*. Elsevier Science, 2003. doi: 10.5555/uri:pii:001650859291831N

[45] A. Mebarki, P. Alliez, and O. Devillers. Farthest point seeding for efficient placement of streamlines. In *Proc. IEEE Visualization*, pp. 479–486, 2005. doi: 10.1109/VISUAL.2005.1532832

[46] B. Moberts, A. Vilanova, and J. J. van Wijk. Evaluation of fiber clustering methods for diffusion tensor imaging. In *Proc. IEEE Visualization*, pp. 65–72, 2005.

[47] Q. Nguyen, P. Eades, and S.-H. Hong. StreamEB: Stream edge bundling. In *Proceedings* 20^{th} *International Conference on Graph Drawing*, pp. 400–413. Springer, 2013. doi: 10.1007/978-3-642-36763-2_36

[48] Q. Nguyen, S.-H. Hong, and P. Eades. TGI-EB: A new framework for edge bundling integrating topology, geometry and importance. In *Proceedings of the* 19^{th} *International Conference on Graph Drawing*, pp. 123–135. Springer, 2012. doi: 10.1007/978-3-642-25878-7_13

[49] U. Ozertem and D. Erdogmus. Locally defined principal curves and surfaces. *Journal of Machine Research Learning*, 12:1249–1286, 2011. doi: 10.1007/s11222-005-4073-8

[50] V. Peysakhovich, C. Hurter, and A. Telea. Attribute-driven edge bundling for general graphs with applications in trail analysis. In *Proc. IEEE Pacific Visualization Symposium (PacificVis)*, pp. 39–46, 2015. doi: 10.1109/PACIFICVIS.2015.7156354

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

[52] L. Piegl and W. Tiller. *The NURBS Book*. Springer, 1997. 2^{nd} edition. doi: 10.1007/978-3-642-97385-7

[53] R. Prado and M. West. *Time Series: Modeling, Computation, and Inference*. Taylor & Francis, 2010. doi: 10.1111/j.1751-5823.2011.00149_6.x

[54] S. Pupyrev, L. Nachmanson, and M. Kaufmann. Improving layered graph layouts with edge bundling. In *Proc.* 10^{th} *International Conference on Graph Drawing*, pp. 329–340. Springer, 2011. doi: 10.1007/978-3-642-18469-7_30

[55] H. Qu, H. Zhou, and Y. Wu. Controllable and progressive edge clustering for large networks. In *Proc.* 5^{th} *International Conference on Graph Drawing*, pp. 399–404. Springer, 2006. doi: 10.1007/978-3-540-70904-6_38

[56] J. Ramsay and B. Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer, 2005. doi: 10.1007/b98888

[57] N. H. Riche, T. Dwyer, B. Lee, and S. Carpendale. Exploring the design space of interactive link curvature in network diagrams. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI)*, pp. 506–513. ACM, 2012. doi: 10.1145/2254556.2254652

[58] W. Saeys, B. De Ketelaere, and P. Darius. Potential applications of functional data analysis in chemometrics. *Journal of Chemometrics*, 22(5):335–344, 2008. doi: 10.1002/cem.1129

[59] R. Scheepens, N. Willems, H. van de Wetering, G. Andrienko, N. Andrienko, and J. J. van Wijk. Composite density maps for multivariate trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2518–2527, 2011. doi: 10.1109/TVCG.2011.181

[60] D. Selassie, B. Heller, and J. Heer. Divided edge bundling for directional network data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2354–2363, 2011. doi: 10.1109/TVCG.2011.190

[61] R. Sibson. SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973. doi: 10.1093/comjnl/16.1.30

[62] H. Stark and J. Woods. *Probability, random processes, and estimation theory for engineers*. Prentice-Hall, 1986. doi: 10.2307/2291115

[63] A. Telea and O. Ersoy. Image-based edge bundles: Simplified visualization of large graphs. *Computer Graphics Forum*, 29(3):543–551, 2010. doi: 10.1111/j.1467-8659.2009.01680.x

[64] C. Tominski, J. Abello, F. van Ham, and H. Schumann. Fisheye tree views and lenses for graph visualization. In *Proc. International Conference on Information Visualisation (IV)*, pp. 17–24, 2006. doi: 10.1109/IV.2006.54

[65] A. Tsai, C. F. Westin, A. O. Hero, and A. S. Willsky. Fiber tract clustering on manifolds with dual rooted-graphs. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–6, 2007. doi: 10.1109/CVPR.2007.383096

[66] M. van der Zwan, V. Codreanu, and A. Telea. CUBu: Universal real-time bundling for large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 22(12):2550–2563, 2016. doi: 10.1109/TVCG.2016.2515611

[67] C.-F. Westin, S. Maier, H. Mamata, A. Nabavi, F. Jolesz, and R. Kikinis. Processing and visualization for diffusion tensor MRI. *Medical Image Analysis*, 6(2):930108, 2002. doi: 10.1016/S1361-8415(02)00053-1

[68] R. Wettel and M. Lanza. Program comprehension through software habitability. In *Proc. IEEE International Conference on Program Comprehension (ICPC)*, pp. 231–240, 2007. doi: 10.1109/ICPC.2007.30

[69] O. W. Winkler. *Interpreting Economic and Social Data: A Foundation of Descriptive Statistics*. Mathematics and Statistics. Springer, 2009. doi: 10.1007/978-3-540-68721-4

[70] N. Wong and S. Carpendale. Supporting interactive graph exploration using edge plucking. In *Proc. SPIE Visualization and Data Analysis*, vol. 6495, pp. 235–246, 2007. doi: 10.1117/12.704612

[71] C.-L. Wu and R. E. Caves. Research review of air traffic management. *Transport Reviews*, 22(1):115–132, 2002. doi: 10.1080/01441640110074773

[72] F. Yao, H.-G. Mller, and J.-L. Wang. Functional linear regression analysis for longitudinal data. *Annals of Statistics*, 33(6):2873–2903, 2005. doi: 10.1214/009053605000000660

[73] H. Yu, C. Wang, C. K. Shene, and J. H. Chen. Hierarchical streamline bundles. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1353–1367, 2012. doi: 10.1109/TVCG.2011.155

[74] H. Zhou, P. Xu, X. Yuan, and H. Qu. Edge bundling in information visualization. *Tsinghua Science and Technology*, 18(2):145–156, 2013. doi: 10.1109/TST.2013.6509098