Scatterplot Summarization by Constructing Fast and Robust Principal Graphs from Skeletons

José Matute* University of Münster
Marcel Fischer † University of Münster
Alexandru C. Telea ‡ University of Groningen
Lars Linsen§ University of Münster

ABSTRACT

Principal curves are a long-standing and well-known method for summarizing large scatterplots. They are defined as self-consistent curves (or curve sets in the more general case) that locally pass through the middle of the scatterplot data. However, computing principal curves that capture well complex scatterplot topologies and are robust to noise is hard and/or slow for large scatterplots. We present a fast and robust approach for computing principal graphs (a generalization of principal curves for more complex topologies) inspired by the similarity to medial descriptors (curves locally centered in a shape). Compared to state-of-the-art methods for computing principal graphs, we outperform these in terms of computational scalability and robustness to noise and resolution. We also demonstrate the advantages of our method over other scatterplot summarization approaches.

1 INTRODUCTION

Scatterplots are one of the main approaches to perform visual data analysis for understanding data distributions and variable relations [6, 15]. They can be created from multidimensional data in various ways, e.g., by selecting two dimensions or by applying dimensionality reduction, or multidimensional projection (MP), methods [38].

For large datasets with complex topologies, or when many scatterplots need to be examined together, such as in scatterplot matrices (SPLOMs), one wants to reduce the cognitive load of the analyst by creating *abstractions* of scatterplots that extract and convey essential aspects of the underlying data. Such known abstractions are density maps [22], principal curves [18], trees [49], graphs [23], and skeletons [28]. Such abstractions can next be used to generate features or characterizations of scatterplots [2] or to create a visual summary of the data [35].

Principal curves (PCs) are one of the oldest of such abstractions [18]. They summarize scatterplots by one-dimensional curves that pass through the "middle" of the data. More formally, PCs are self-consistent curves: The location of any PC point coincides with the average of all scatterplot points that project there. Principal curves have been used in many applications. In particular, they are part of the original scagnostics characterizations proposed by Tukey and Tukey [45]. They are also used for character recognition [23], ice floe identification [1], and feature extraction and classification [4]. *Principal graphs* (PGs) generalize PCs to sets of connected self-consistent curves that extend the summarization power of PCs to more complex scatterplot topologies and geometries.

Computing PCs and PGs is not trivial. Limitations of current methods include long computation time, convergence rates that depend on proper initialization, sensitivity to noise (spurious scatterplot points), and no easy control of the summarization's level of detail. We propose a new approach for computing PCs and PGs based on medial descriptors (Sec. 3). Our main contributions can be summarized as: *(C1) fast and robust computation of principle graphs*: Our approach is one to two orders of magnitude faster than state-of-the-art methods, robust to noise, can treat any scatterplot topology, is simple to use, and allows one to specify the summarization's level of detail; *(C2) structured summarization of scatterplots:* the summarization is stored as a graph structure, allowing for furcation analysis and visual summarization of local density and deviations. Additionally, we propose an extended Hausdorff distance for PG quality comparisons. To our knowledge, this is the first time when PGs are compared quantitatively. In contrast to recent work in scatterplot summarization [28] that also uses skeletons for summarization, the method we present here computes PGs following their accepted formal definition [18], whereas in [28] a different, simpler descriptor is computed that ignores scatterplot density variations (cf. Fig. 11, Sec. 4.4).

To demonstrate the advantages of our method, we compare it quantitatively to state-of-the-art PG construction approaches and also to ground-truth data-generating curves. We also compare our method qualitatively to other graph-like scatterplot abstraction methods and outline the advantages of our approach in this regard (Sec. 4). Finally, we show how our approach can be used for the visual analysis of multidimensional data (Sec. 5).

2 RELATED WORK

Let $D = \{\mathbf{p}_i\} \subset \mathbb{R}^d$, $1 \leq i \leq N$, be a d-dimensional dataset with N points $\mathbf{p}_i$. Let $S = \{\mathbf{x}_i\} \subset \mathbb{R}^2$ be a scatterplot generated from D, where point $\mathbf{x}_i$ maps the data point $\mathbf{p}_i$. We organize related work into approaches for computing principal curves and graphs, approaches for abstracting scatterplots, and approaches for abstracting shapes.

2.1 Principal curves and graphs

Hastie and Stuetzle introduced principal curves (PCs) as 1D curves that pass through the "middle" of the data [18]. More formally, PCs fulfill the self-consistency criterion, i.e., the average of all scatterplot points that project to a PC point is equal to that point. PCs are constructed from a polygonal line, initialized to the scatterplot's first principal component, whose vertices are iteratively optimized using a projection and a conditional expectation step. This method, however, is computationally inefficient and cannot handle multiple trends in a scatterplot.

Banfield and Raftery [1] extended the original approach to handle closed curves and to reduce bias using a penalty term. Tibshirani [44] also tried to minimize bias by redefining PCs in terms of a mixture model and conditionally independent distributions over the PC points. Kégl et al. [24] provide a regularized version of PCs based on the self-consistency criterion, where, given a bounded length, the PC always exists. This approach uses iterative vertex creation-and-optimization steps from an initial line based on distributions over Voronoi regions of a polygonal line. We follow this idea to exploit the connection between the points' projection to the PC and the Voronoi regions. Our approach can also handle closed curves.

Several other approaches estimate PCs using differing criteria [3, 5, 10]. The only approach we know of that allows for PC self-intersections is the one by Verbeek et al. [47]. They also used Voronoi regions but optimized line segments instead of vertices, which are

*e-mail: matutefl@uni-muenster.de
†e-mail: m_fisc35@uni-muenster.de
‡e-mail: a.c.telea@rug.nl
§e-mail: linsen@uni-muenster.de

subsequently connected. The resulting polyline segments only yield a coarse PC approximation, which could be used to initialize other PC algorithms [18].

Kégl and Kryzak [23] extended their polygonal line PC algorithm to compute skeletons of binary shapes with applications in character recognition. An initial graph is defined by thinning the input shapes. The resulting vertices are optimized (displaced or deleted) according to a number of angle criteria. While this method can handle complex topologies, it does not actually compute PGs of *arbitrary density* scatterplots, but only of binary shapes, which are a particular case of uniformly and equally densely-sampled scatterplots. Özertem and Erdogmus [30, 31] propose a subspace-constrained mean-shift algorithm where points are moved following the local gradient and local Hessian of the probability density function describing the scatterplot, which is computed using kernel density estimation (KDE). The efficiency of this approach highly depends on the number of points and amount of noise in the data. Our proposed approach also allows for computing PGs. We compare it to both aforementioned approaches [23, 30, 31] and show its higher robustness and speed.

2.2 Scatterplot abstraction

Scatterplot abstractions are used to generate features and/or metrics to characterize a scatterplot [2] or to create a visual summary of the data [35].

Tukey and Tukey [45] proposed to characterize scatterplots by several so-called *scagnostics* (scatterplot diagnostics) measures. These include measures from geometric graph analysis, measures computed from PCs, and kernel-based measures. To increase the efficiency of computing such measures, Wilkinson et al. [49] defined graph-theoretic characterization measures based on the geometric characteristics of a minimum spanning tree (MST) of the binned point distribution of S. Although these measures are much faster to compute than the original ones [45], extracting scagnostics from large scatterplots is still seen as computationally challenging [9].

Reddy et al. [34] extended PCs to so-called data skeletons or principal trees which summarize scatterplots where multiple trends occur. They partition the dataset D into k clusters and compute a minimum spanning tree (MST) from their centroids. D is then filtered to contain only clusters traversed by the MST from one endpoint to another. For m such endpoints, $m(m-1)/2$ PCs are computed, which is quite computationally intensive. Also, if k is not chosen appropriately, the summarization does not capture well the shape of S.

Gerber et al. [16] proposed a simplified geometric representation of high-dimensional data based on regression curves and dimensionality reduction. The set of regression curves represents a topology-based skeleton of the data D. While not directly applicable to 2D scatterplots $S \subset \mathbb{R}^2$, this method can be used to simplify density maps computed from scatterplots.

Visual summarization can cover more than the position of scatterplot points. At point level ($\mathbf{x}_i \in S$), one can encode information about the dimensions of $\mathbf{p}_i$ in color, opacity, shading [8], or glyph shape. This works well for scatterplots of under thousand points or when subsampling S to create more visual space to encode data for groups of related points, e.g., via tag clouds [33]. In large scatterplots, overplotting occurs so attributes of individual points cannot be distinguished anymore. At scatterplot level, this can be mitigated by aggregating values via density maps [27, 36]. At an even higher aggregation level, functional boxplots, computed by binning over data ranges, show summary statistics [39]. For large SPLOMs depicting high-dimensional datasets, a focus-plus-context approach is used [50], as individual scatterplots become too small to understand. We propose to summarize scatterplots using their PGs enriched to visually encode density and deviation of the data.

2.3 Shape abstraction

Scatterplots can be seen as a particular form of 2D shapes. While they are, formally speaking, not compact subsets of $\mathbb{R}^2$ (as shapes are), for a large point count and when rendered on small target areas (like in large SPLOMs), scatterplots converge in the limit to compact shapes. Moreover, humans perceive scatterplots and shapes by means of the same type of visual features, e.g., size, skewness, orientation, genus, curvature, and thickness [32]. Hence, shape abstraction techniques in computer vision are also relevant for scatterplot characterization.

Medial axes, or *skeletons*, are one such powerful descriptor [37]. For a 2D shape $\Omega \subset \mathbb{R}^2$ with boundary $\partial\Omega$, let DT_Ω be its so-called distance transform [7], given by

$$DT_\Omega(\mathbf{x} \in \mathbb{R}^2) = \min_{\mathbf{y} \in \partial\Omega} \|\mathbf{x} - \mathbf{y}\|. \quad (1)$$

The skeleton of Ω is defined as

$$S_\Omega = \{\mathbf{x} \in \Omega | \exists \mathbf{f}_1 \in \partial\Omega, \mathbf{f}_2 \in \partial\Omega, \mathbf{f}_1 \neq \mathbf{f}_2, \|\mathbf{x} - \mathbf{f}_1\| = \|\mathbf{x} - \mathbf{f}_2\| = DT_\Omega(\mathbf{x})\} \quad (2)$$

The points $\mathbf{f}_i$, called *feature points* of skeleton point $\mathbf{x}$, are the closest points on $\partial\Omega$ to $\mathbf{x}$ and define the *feature transform* FT_Ω of Ω [20]. Computing S_Ω by directly applying Eqn. 2 is delicate, since small details over $\partial\Omega$ can create large changes, also called spurious branches, in S_Ω [37]. To address this, one computes *regularized* skeletons $S^\tau_\Omega = \{\mathbf{x} \in S_\Omega | \gamma(\mathbf{x}) \geq \tau\}$. Here, $\gamma : S \to \mathbb{R}^+$ is a so-called importance metric, low over spurious branches and high elsewhere on S_Ω. A well-known such metric sets $\gamma(\mathbf{x})$ to the shortest path along $\partial\Omega$ between the feature points $\mathbf{f}_1$ and $\mathbf{f}_2$ of $\mathbf{x}$ [14, 40, 41, 43]. Hence, S^τ_Ω is a multiscale skeleton in which all branches of S_Ω caused by boundary details shorter than τ length-units have been removed. Multiscale skeletons of shapes sampled as images of megapixel resolution can be computed in subsecond time on the CPU [43] and milliseconds on the GPU [13].

Besides simplifying binary shapes, skeletons have also been used to summarize scatterplots by Matute et al. [28]. They show that their measures based on medial axes outperform earlier scagnostics measures in terms of closeness to perceptually-based similarity and computational efficiency. The resulting skeletons were defined as point-sets where the structure of the scatterplot is not maintained. Our proposed principal graph maintains the information of the topology of the data defined as node-link structure. Moreover, and similar to Wilkinson et al. [49], their approach ignores the local density variations in a scatterplot. Thus, it does *not* compute principal curves or graphs in the sense of any of the definitions above [18, 23, 30, 31], but simply a locally-centered one-dimensional summarization of the scatterplot shape. The main motivation behind this simple, but density-ignoring summarization was that it was much faster to compute than 'true' density-aware PG methods. The approach we propose computes true PGs, but at a fraction of the cost of existing methods. We will show later, cf. Figure 11 in Sec. 4.4, an example of the differences between a true, density-centered PG and the simpler skeleton summarization proposed in [28]. In a different context, skeletons have been used to summarize large graphs and trail-sets by bundling [13, 42]. This approach shares the limitations of [28], i.e., the local edge density is not considered during summarization.

3 METHOD

Our method follows the iterative principle used by most existing PC techniques, which we outline in Section 3.1. We then detail our method to generate PGs in Sections 3.2-3.4. Finally, we augment our PG summarizations with encodings of local density and standard deviation of the data in Section 3.5.

3.1 Iterative principal curve construction

The original approach for computing PCs [18], and up to large extents all follow-up approaches [1, 23, 47], iteratively adapts an explicit polyline-based representation f of the PC to satisfy the self-consistency criterion given a scatterplot S. First, f is initialized with an estimate f^0 of the PC. Finding a good estimate f^0 is important to reduce the number of iterations and crucial to yield converging PCs. Then, in each iteration the following two steps are executed:

Projection step: Assuming a 1D parametric representation $f^j(\lambda) \subset \mathbb{R}^2$ of the PC after j iterations, the points $\mathbf{x}_i \in S$ are projected onto $f^j(\lambda)$. For this, one evaluates a so-called projection index $\lambda_{f^j}(\mathbf{x})$ [18], i.e., the value of λ for which $f^j(\lambda)$ is closest to $\mathbf{x}$, as

$$\lambda_{f^j}(\mathbf{x}) = \sup_{\lambda}\{\lambda : \|\mathbf{x} - f^j(\lambda)\| = \inf_{\mu}\|\mathbf{x} - f^j(\mu)\|\} \; . \qquad (3)$$

Conditional expectation step: A new curve f^{j+1} is estimated from the current f^j by shifting points $\mathbf{y} \in f^j$ towards the average of all points that project to $\mathbf{y}$. The process repeats until f^j converges. For details, we refer to Fig. 3 in [18] and related text.

3.2 Skeleton-based principal graph construction

Our method, to which we refer as Skeleton-based Principal Graph (SPG), is based on the following key observation: Let $Z \subset \mathbb{R}^2$ be a compact 2D subset that covers a subset of points $S_Z \subset S$ of a scatterplot S. If points in S_Z are uniformly distributed over Z, then the PC of S_Z coincides with the medial skeleton of S_Z. For non-uniform distributions, the PC deviates from the skeleton of S_Z in the sense that it is "pulled" away towards the higher-density areas of Z rather than staying in its geometric center. Hence, skeletons and PCs are closely related.

Our method takes advantage of this relationship, as follows (see also pseudocode in Alg. 1): We start by creating a scatterplot S from a given high-dimensional dataset D using any suitable method, e.g., dimension selection or multidimensional projection (line 1). Next, we transform S to a compact shape Ω and extract its skeleton S_Ω (lines 2-4, Sec. 3.3). We then use S_Ω to initialize our principal graph G (line 5) and perform the projection and conditional expectation steps mentioned in Sec. 3.1 (lines 6-12). Since S_Ω is a good approximation of the PG, the process converges after a few steps. Finally, we optionally augment the resulting principal graph G to visually summarize local density and standard deviation of the data (Sec. 3.5). In summary, SPG consists of a skeleton construction step (with parameters $\sigma_{SPG}, \tau_{SPG}, r_{SPG}$, see Sec. 3.3) and an iterative principal graph construction (with parameters m_{SPG}, g_{SPG}, see Sec. 3.4).

Algorithm 1 Skeleton-based principal graph construction

Require: D, σ_{SPG},r_{SPG},τ_{SPG},m_{SPG},g_{SPG}
1: $S \leftarrow projection(D)$
2: $I \leftarrow densityEstimation(S, r_{SPG}, \sigma_{SPG})$
3: $\Omega \leftarrow binaryShape(I)$
4: $S^\tau_\Omega \leftarrow skeletonization(\Omega, \tau_{SPG})$
5: $G \leftarrow initializeGraph(S^\tau_\Omega, m_{SPG})$
6: **while** $max\|\hat{c}_i - c_i\| \geq \varepsilon$ **do**
7: **for all** c_i in G **do**
8: $VN \leftarrow neighborVoronoiCells(c_i, g_{SPG})$
9: $c_g = centroid(\mathbf{x} \in S \cap VN)$
10: $\hat{c}_i \leftarrow \frac{c_g + c_i}{2}$
11: **end for**
12: **end while**

3.3 Skeleton construction

To construct a skeleton, we need a *compact* 2D shape Ω embedded in $\mathbb{R}^2$. We construct such a shape from a scatterplot S as follows. First, we discretize (bin) S to a grayscale image with $r_{SPG} \times r_{SPG}$ pixels , i.e., the desired resolution. Next, we compute the discrete kernel density estimation (KDE) $I(\mathbf{x})$ of S for $\mathbf{x} \in [1, r_{SPG}]^2$ by

$$I(\mathbf{x}) = \sum_{\mathbf{y} \in S} K\left(\frac{\|\mathbf{x} - \mathbf{y}\|}{R}\right) \qquad (4)$$

where K is a Gaussian kernel of standard deviation σ_{SPG} pixels, stored as a $R \times R$ pixels grayscale texture, where $R = 3\sigma_{SPG} + 1$ ensures a proper kernel discretization [17]. Next, we obtain a compact binary shape Ω by upper thresholding I by the average density $\hat{\rho} = N/(r_{SPG})^2$ of the scatterplot S. Figure 1 shows the shape Ω for the *Spiral* dataset from [25]. Thesholding I at higher density values preserves only denser scatterplot regions and eliminates (possibly spurious) low-density details; lower thresholds capture more of S. That is, points in S farther away from their nearest neighbors than σ_{SPG} are filtered out. This provides a simple but robust way to capture, or *summarize*, the essence of S into a 2D shape.

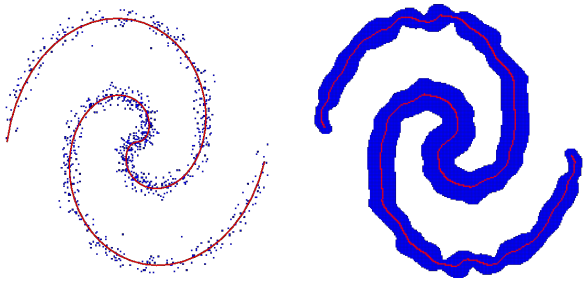


Figure 1: Skeleton generation for the *Spiral* dataset. The scatterplot (left) is convolved with a Gaussian kernel, and a compact shape (blue in right image) is obtained by thresholding the expected average density. The skeleton of this shape is shown in red in the right image.

Having Ω, we now compute its distance transform and skeleton (following Eqns. 1, 2) using the Augmented Fast Marching Method (AFMM) [43]. Next, we regularize S_Ω using the boundary-length importance metric γ (Sec. 2.3). Following [43] , we set τ to roughly 5% of the boundary length $\|\partial\Omega\|$, which yields a good balance between keeping details of the scatterplot's shape and not creating overly detailed spurious branches.

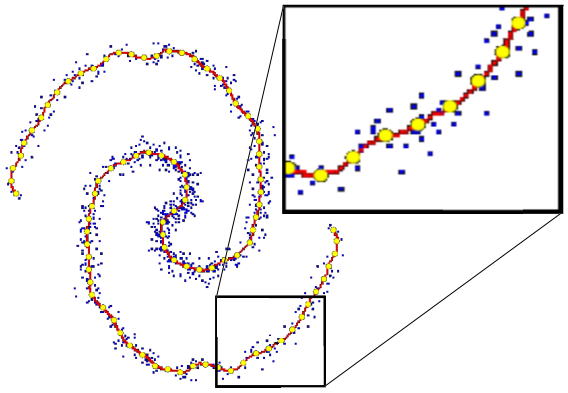


Figure 2: PG generation from skeleton of *Spiral* dataset. The image and inset show graph vertices or control points (yellow), graph edges (red), and original scatterplot points (blue).

3.4 Iterative principal graph construction

We generate the initial estimate of the principal graph G from the skeleton $S^\tau(\Omega)$. Before doing this, we perform a topologically stable

one-pass thinning step on $S^\tau(\Omega)$. This ensures that the skeleton we next consider to generate G is always one pixel thick, which makes its conversion into a graph simpler. To generate the vertices of G, we sample the 1D curve set $S^\tau(\Omega)$ in arc-length space using a sampling distance m_{SPG}, i.e., two consecutive vertices, linked by an edge in G, are at a distance m_{SPG} along the pixel-chain that defines $S^\tau(\Omega)$. We start sampling from the endpoints of the branches of $S^\tau(\Omega)$ or, when $S^\tau(\Omega)$ consists only of loops, from a randomly selected point in it. The initial estimate G is then the straight-line drawing of this graph, i.e., with edges rendered as line segments. Figure 2 shows the graph G for the *Spiral* dataset shown earlier in Fig. 1.

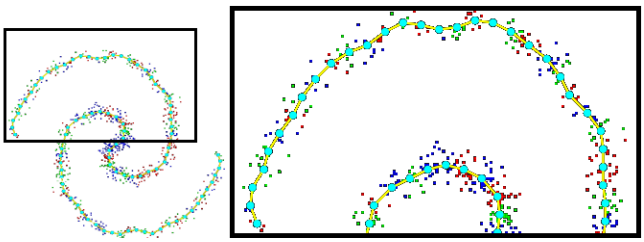


Figure 3: Closest points (small dots) to a set of control points (large dots) for *Spiral* dataset. Color indicates which scatterplot points are associated to which control points.

Having the graph G, we now implement the two PC computation steps (Sec. 3.1) for PGs as follows.

Projection step: Let $C = \{c_1, \ldots, c_n\}$ be the n control points (vertices) of G. Consider the 2D Voronoi diagram $V(C)$ having C as sites. Cell $V(c_i)$ of $V(C)$ contains all pixels closer to c_i than to any other control point in C. Due to its relationship to Eqn. 2, computing $V(C)$ is straight forward: We run the AFMM skeletonization algorithm (Sec. 3.3) with the set of pixels containing the control points C as input shape. The resulting feature transform $FT_C(\mathbf{x})$, evaluated at any pixel $\mathbf{x}$, provides the control point in C closest to $\mathbf{x}$. Figure 3 shows this: Large dots are control points in C. Small dots are the scatterplot points, colored to indicate their closest control point. As a side note, we can also see here the analogy between the distance transform $DT_C(\mathbf{x})$ and the projection index $\lambda_f(\mathbf{x})$ (Eqn. 3). The relationship between Voronoi diagrams and the projection of scatterplot points on the PC had been pointed out in earlier work [23,24,47], but our usage of image-based Voronoi diagrams to define PGs is novel. Furthermore, we note that earlier methods to compute image-based Voronoi diagrams exist such as based on splatting the radial profile of a one-point DT encoded as a polygon mesh [21] or as a luminance texture . However, such methods are significantly slower, as splatting causes a large amount of overdraw (for the description of the problem, see the critique of [22] in [13]), or use a coarse DT sampling, which decreases accuracy. In contrast, our method is pixel-accurate and runs in $O(N \log N)$ for an image of N pixels.

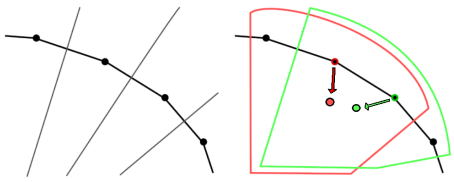


Figure 4: (left) Voronoi cells for graph G. (right) Union VN of neighboring Voronoi cells that affect two selected control points marked red and green, respectively, for $g_{SPG} = 1$. Control points move in the direction of their respective centroids.

Conditional expectation step: Using the feature transform FT_C, we can find, for any control point $c_i \in C$, the set of scatterplot points closest to c_i as $N(c_i) = \{\mathbf{x} \in S | FT_C(\mathbf{x}) = c_i\}$. Then, we simply move the control point c_i half-way towards the average of $N(c_i)$, yielding an updated control point $\hat{c}_i$. Note that c_i are *not* constrained to the image grid, i.e., the control points have all freedom to move to best approximate the PG.

We repeat the projection and conditional expectation steps until C converges, i.e., until $max\|\hat{c}_i - c_i\| \leq \varepsilon$, where ε is set to one pixel. This is similar to applying Lloyd's method [26] for constructing weighted Voronoi diagrams – though, in our case, we do not have explicit cell weights we want to realize. A similar relationship between k-means and the conditional expectation step was proposed earlier [47], where k-means was extended to k-line segments. Our work is different, as we do not use k-means.

Directly applying the update rules above can result to instability, as even small changes to the positions of c_i may significantly change the shapes of their corresponding Voronoi cells $V(c_i)$, a well-known "duality" to the skeletal instability mentioned in Sec. 2.3. To address this, we move the control point c_i to the average of all scatterplot points in the union VN of all Voronoi cells $V(c_j)$ whose centers c_j have at most topological distance $g_{SPG} \in \mathbb{N}$ from c_i on the graph G (Fig. 4). For control points c_i which have a single neighbor in G, i.e., endpoints of paths in G, we always set $g_{SPG} = 0$ such that these points do not drift inwards. Overall, g_{SPG} has the effect of a smoothing low-pass filter that removes unwanted oscillations from an iterative converging process. Using $g_{SPG} = 1$ yields the desired smoothness and stability. For similar examples, though in the different context of edge bundling, we refer to literature [13, 46]. If $g_{SPG} = 0$, the control points for the generated Voronoi diagrams correspond to the respective cells' mass-centers, i.e., our computation yields centroidal Voronoi tessellations [12].

Our algorithm converges in a few iterations (less than 10 in all tested cases). Key to this process is the initialization of the PG with the (sampled) medial skeleton which, as explained above, is already quite close to the desired PG, modulo accounting for the variation in scatterplot point density. Figure 5 details this: Here, we show the so-called generating curve (green) used to create the scatterplot by spreading points around it [18,24], the skeleton initializing the PG generation (red), and the final PG (yellow). We see that all three curves are very close to each other. It is important to stress that no other PC method we know of has a comparably good initialization.

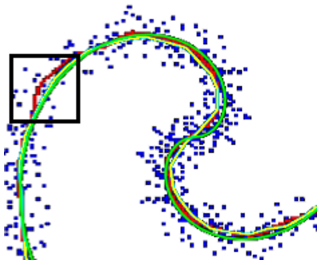


Figure 5: Generating curve (green) for scatterplot (blue), skeleton (red), and computed principal curve (yellow) for *Spiral* dataset. We observe very good agreement between the three curves. Small variations in shape may result in diverging curves for skeleton-based summarizations (black frame) [28].

3.5 Visual summarization of local properties

We propose to optionally enhance the summarization delivered by the PG G of a scatterplot S by visually encoding local density and standard deviation. For each control point c_i of PG G, we generate local statistics within the respective Voronoi cell $V(c_i)$.

We define a line l_i that goes through control point c_i and is orthogonal to the tangent to PG G at c_i. We project all scatterplot points within Voronoi cell $V(c_i)$ to line l_i. Then, we can create 1D summary statistics within l_i such as computing the standard deviation σ_i from

c_i. To define the tangent we have to consider three cases: If the degree of c_i within G is 1, i.e., c_i is an endpoint, then the tangent is the line that contains the only graph edge that connects c_i. If the degree of c_i is 2, then the tangent is defined as the line that contains c_i and has the same (minimum) angle to the two edges connecting c_i. If the degree of c_i is larger than 2, i.e., we have a branching, then we combine multiple degree 2-case by taking the average direction of the tangents. To visually encode the standard deviation, we encapsulate the PG with bands, where the band width at control points c_i to both sides encodes σ_i. This visual encoding is inspired by the bands in curve box plots [29], which is a generalization of boxplots to curves.

Moreover, we associate a local density ρ_i with control point c_i, which we define as the number of points within Voronoi cell $V(c_i)$ divided by the overall number of scatterplot points N and the average length of the edges connected to c_i. The local density is visually encoded by the opacity of the bands. Figure 6 provides the PG summarization for the *Spiral* dataset augmented with the band drawing. By construction, this dataset has similar standard deviation and density throughout, which can be observed from the band's (almost) constant width and opacity.

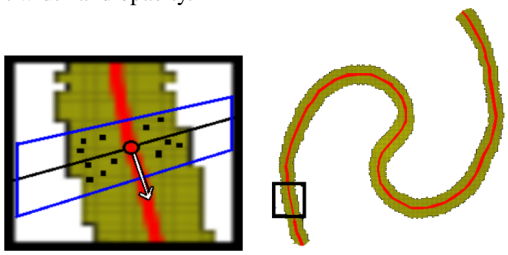


Figure 6: Visual summarization of the scatterplot of the *Spiral* dataset. The PG (red) summarizes shape and the surrounding band (yellow) summarizes standard deviation (width) and local density (opacity), which are both (almost) constant for this dataset.

4 Evaluation

We evaluate our approach (SPG) with regard to noise, scatterplot size, and resolution of the used images by comparing against the two state-of-the-art methods that can handle PGs with intersections and loops, namely, Kégl's Principal Graphs (KPG) [23] and KDE-based Subspace Constrained Mean Shift (KDE-SCMS) [31]. Both implementations are available online at the respective authors' websites. We did not consider comparing to other methods [18,47], since these are either obviously not handling curves having intersections and/or loops or have been outperformed by the approaches we compare against. Atop of the above, we provide a comparison to two state-of-the-art approaches to characterize scatterplots by their shape, namely Graph-Theoretic Scagnostics [49], implemented in R [48] and Skeleton-based Scagnostics [28]. A more detailed overview of the methods is provided below.

Name	Function	No. Points	Noise Range	Noise step
S(1-7)	Spiral	1000	0.075 - 0.3	0.015
S(8-14)	Spiral	5000	0.075 - 0.3	0.015
S(15-21)	Spiral	10000	0.075 - 0.3	0.015
H(1-5)	Helix	1000	0.15 - 0.3	0.0375
H(6-10)	Helix	5000	0.15 - 0.3	0.0375
H(11-15)	Helix	10000	0.15 - 0.3	0.0375
R(1-5)	Rune	5000	0.075 - 0.225	0.0375
R(6-10)	Rune	7500	0.075 - 0.225	0.0375
R(11-15)	Rune	10000	0.075 - 0.225	0.0375

Table 1: Datasets with varying degrees of complexity, noise, and sizes, used in our evaluation.

4.1 Compared methods

Kégl's Principal Graphs (KPG): Kégl and Krzyzak [23] proposed a principal graph algorithm for piecewise skeletonization. The method is related to ours as it uses an image-based approach, as well as medial skeletons. In contrast to our skeletons [43], they compute skeletons by a (less accurate) thinning approach. As in our case, the skeletal graph is fitted and smoothed to converge to a PG using various vertex optimization steps. This method reads a binary image as input, therefore we directly fed it with our binary shape Ω constructed by KDE (Sec. 2.3).
Subspace Constrained Mean Shift (KDE-SCMS): Özertem and Erdogmus [31] proposed a subspace-constrained mean-shift method where scatterplot points are shifted using the local gradient and Hessian of the data's probability density function found via kernel density estimation (KDE), using a Gaussian kernel, like we do. The local gradient and Hessian are computed at each iteration.
Graph-Theoretic Scagnostics: Wilkinson et al. [49] defined nine features to characterize scatterplots, using the scatterplot's nearest-neighbor-graph's minimum spanning tree (MST) and Spearman's correlation coefficient. To increase efficiency, the MST is constructed from binned data rather than the entire scatterplot.
Skeleton-based Scagnostics: Matute et al. [28] defined two dissimilarity measures based on the adjacency graph formed by skeletons constructed from scatterplots in much the same way we use ourselves to initialize the PG construction (Sec. 2.3). These metrics are then used to gauge how different, or similar, various scatterplots are.

4.2 Evaluation datasets

To compare the methods, we chose to use synthetic datasets. The key reason is that, for such data, we have *ground truth*, i.e., we know what the summarization should be. We used three generator curves with and without loops, distribute points uniformly around them, and also add Gaussian spatial noise (Fig. 7 a-c). A good PG algorithm should extract curves that are close to these generators. Table 1 summarizes the generated datasets, also available as supplementary material. The three tested PG methods (Sec. 4.1) were each run 20 times per dataset, after which we recorded their average performance. Figure 7 shows examples of the PGs computed by each method. The datasets are detailed next.
Spiral: The generating curve for this dataset is given by

$$x = -\cos(t) * t, \quad y = \sin(t) * t$$

with t uniformly distributed in $[0, 2\pi]$, and noise added independently over both x and y dimensions with a normal distribution $\mu = 0$. For our method (SPG), we used the parameters $\sigma_{SPG} = 4$, $\tau_{SPG} = 15$, $m_{SPG} = 6$, and $g_{SPG} = 1$, and three resolutions $r_{SPG} \in \{128, 192, 256\}$ pixels squared. We used the same resolutions for KPG. For KDE-SCMS, we used $\sigma = 0.5$. The same σ was used to generate the binary input shape Ω.
Helix: The generating curve is given by

$$x = t, \quad y = \pm(1.0 + t) * cos(t)$$

with t uniformly distributed in $[0, 3\pi]$, and noise added independently for both x and y dimensions with a normal distribution with $\mu = 0$. For our method, we used $\sigma_{SPG} = 3$, $\tau_{SPG} = 10$, $m_{SPG} = 6$, and $g_{SPG} = 1$ for the three tested resolutions. For KDE-SCMS, $\sigma = 0.5$ was used – the same value used to generate the binary shape Ω. For KPG, we used its default parameters and varied only the image resolution.
Rune: The generating curves (concentric circles) are given by

$$x = r * cos(t), \quad y = r * sin(t)$$

with $r \in \{1.5, 3.0, 4.5\}$ and t is uniformly distributed in $\{[0, 2\pi], [0.4\pi, 1.66\pi], [0.66\pi, 1.5\pi]\}$ for each level respectively. The radial lines are defined by setting t to 0.25π and 1.25π and r as

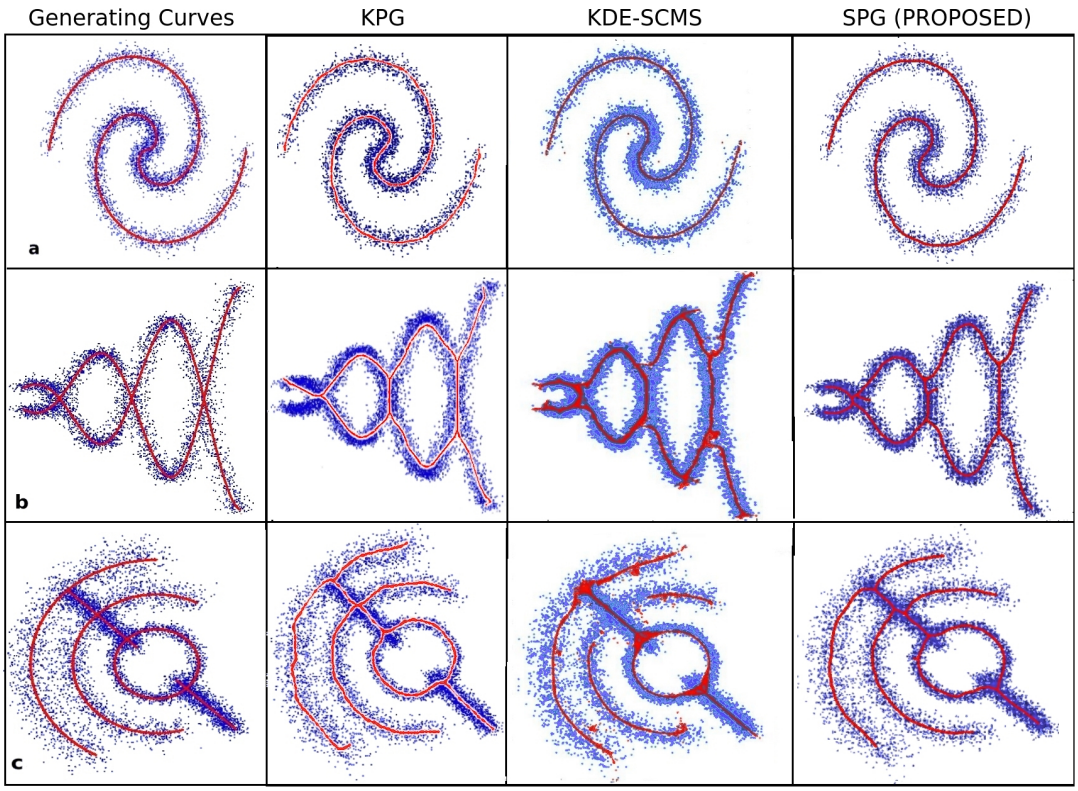


Figure 7: Principal graphs for three datasets of increasing complexity (self-intersections and loops). (a) *Spiral*. (b) *Helix*. (c) *Rune*. Our method (SPG) tested against two state-of-the-art methods (KPG, KDE-SCMS) as well as the data-generating curves.

$1.0+s$, with s uniformly distributed in $[0,3]$. Noise is independently distributed in the x and y dimensions like for the *Spiral* and *Helix* datasets. For our method, we use $\sigma_{SPG}=3$, $\tau_{SPG}=25$, $g_{SPG}=1$, and $m_{SPG}=6$. For KDE-SCMS we use $\sigma=0.5$. For KPG, we generated the shapes Ω like for the *Spiral* and *Helix* datasets.

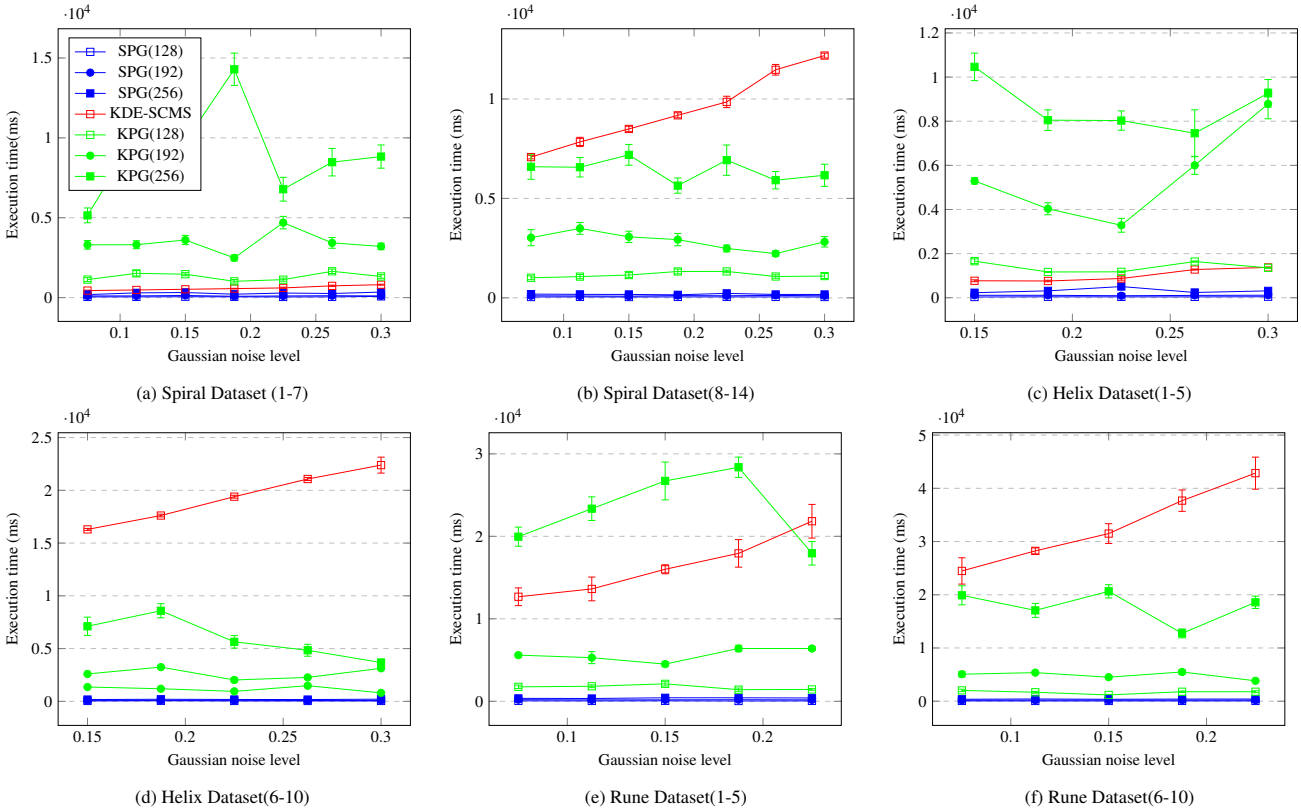
4.3 Computational time comparison

Spiral: For a low point count, KDE-SCMS has similar speed to our method (Fig. 8a). For larger point counts, our method becomes much faster (Fig. 8b): For 10K points, noise=0.075 (S15 dataset), KDE-SCMS takes 25160 ± 190 ms while our method takes $48.4 \pm$ 1ms, i.e., *over 600 times faster*. KPG is even slower than KDE-SCMS for low point counts. The quite drastic change in execution times *vs* noise levels is caused by the smoothness of the compact shape Ω – the smoother Ω is, the fewer branches it skeleton has, so a smaller execution time ensues. For large scatterplots, timings reverse for KDE-SCMS and KPG (Fig. 8b), due to the compact shape used in KPG's implementation. KPG and SPG both outperform KDE-SCMS for large point counts. Yet, SPG is one to two orders of magnitude faster than KPG.

Helix: As for *Spiral*, the speed of KDE-SCMS and SPG are quite similar for low point counts (Fig. 8c). KPG is slower than both approaches regardless of resolution. The speed variance of KPG depends on the shape of the compact image it gets as input. For smooth boundaries, a simpler skeleton is created with few vertices; non-smooth boundaries cause spurious branches to be trimmed by KPG's rule set. For higher point counts (dataset H(6-10), Fig. 8d), our approach clearly outperforms KPG and KDE-SCMS speed-wise, as for the *Spiral* dataset. KPG's speed depends only on the default parameters and image resolution, so similar execution times are achieved as in Fig. 8c. For higher point counts (H11 dataset), our approach takes 38 ± 8.2 ms; KDE-SCMS takes $4.6 \times 10^4 \pm 4 \times 10^3$ ms; and KPG takes 1519 ± 34 ms for the lowest considered resolution. For the highest resolution and noise level (H15 dataset), our approach takes 180.5 ± 20 ms; KDE-SCMS takes $6849 \times 10^4 \pm 5.07 \times 10^3$ ms, and KPG takes 8193 ± 740 ms respectively. Again, SPG is one to two orders of magnitude faster.

Rune: We get similar timing patterns for low point counts (Fig. 8e): KPG is slower on low point counts; KDE-SCMS gets slower with noise level. We find again this trend reversed for higher point counts (H(6-10) dataset, Fig. 8f). At the lowest point count, noise, and resolution (R1 dataset), SPG takes 53 ± 4.8 ms; KDE-SCMS takes 12673 ± 1076 ms; and KPG takes 1752 ± 214.8 ms, respectively. SPG is again about two orders of magnitude faster. For the most complex dataset tested (R15), the difference is similar: SPG takes 375.4 ± 3.6 ms; KDE-SCMS takes 62824 ± 729.5 ms; and KPG takes 20285 ± 1625 ms.

Our method is fast for two main reasons. First, our *initialization* by the AFMM regularized skeleton gives us an already very good PG estimate. Thus, we only need few (about 10) iterations to converge. In contrast, KDE-SCMS needs substantially more iterations. KPG, on the other hand, starts from a far less accurate skeleton (produced by thinning), which requires several complex cleaning steps. Secondly, our *update* step is very simple by just shifting a control point c_i to the average location of its neighbors $N(c_i)$ in the scatterplot S. Finding the neighbors $N(c_i)$ is very fast (using FT_C, see Sec. 3.4). In contrast, KPG needs to shift *all* points in S at every iteration, and there are far more points in S than on the skeleton (which yields the control-point set C). Furthermore, KDE-SCMS uses a numerical optimization (gradient descent) *at each iteration*, which is obviously more costly than the simple shift-to-average operation we are performing. More formally, for a scatterplot of $|S|$ points, an image of $N=r_{SPG}^2$ pixels, and a kernel K of R^2 pixels (where by definition $R \leq r_{SPG}$), the asymptotic costs of our method is $O(|S| \cdot N)$ for computing the density



(a) Spiral Dataset (1-7) (b) Spiral Dataset(8-14) (c) Helix Dataset(1-5)

(d) Helix Dataset(6-10) (e) Rune Dataset(1-5) (f) Rune Dataset(6-10)

Figure 8: Computation time for principal graphs for *Spiral*, *Helix*, and *Rune* datasets with varying number of points, noise levels, and resolutions $r_{SPG} \in \{128, 192, 256\}$. Our approach outperforms KDE-SCMS and KPG in terms of computational speed.

map, $O(N \log N)$ for computing the skeleton and associated quantities, and $O(I \cdot N \log N)$ for computing the I conditional expectation steps (since computing $V(C)$ has the same cost as skeletonization). Since, in practice, one uses a high but fixed image resolution N, the overall cost is linear in the number of scatterplot points $|S|$.

4.4 Quality evaluation

Visual comparison: As outlined already, PGs should be very close to the generating curve of a dataset, with the possible exception of high-curvature areas. Figure 7 compares this visually. We see that the KPG results degrade with increasing amount of noise: For the Helix dataset, one branch was completely pruned due to KPG's rule set; for the Rune dataset, the PG curves are rather bumpy. KDE-SCMS has issues with the choice of σ: For the Rune dataset, some areas become a blob rather than a curve due to plateau convergence, while in other areas the curve structure is missed already. In contrast, SPG produces desired and expected results.

Quantitative comparison: To quantify quality, we measure the difference between PGs and the ground-truth generating curve. For this, we adapt the well-known Hausdorff distance [19]: Given two PGs G_1 and G_2, we define their distance $D(G_1, G_2) \in \mathbb{R}^+$ as

$$D(G_1, G_2) \quad = \quad \max\left(\max_{\mathbf{x} \in G_1}(DT_{G_2}(\mathbf{x})), \max_{\mathbf{x} \in G_2}(DT_{G_1}(\mathbf{x}))\right). \quad (5)$$

We found a median distance D of ≤ 0.10 when comparing our extracted PGs with the generating curves for datasets S(1-21), H(1-15), and R(1-15). The mean distances were 0.0491 ± 0.029 (*Spiral*), 0.0546 ± 0.0152 (*Helix*), and 0.089 ± 0.039 (*Rune*). The 25%,50% and 75% distances were 0.016, 0.042 and 0.088 (S(1-21) dataset); 0.047, 0.052, and 0.066 (H(1-15) dataset); and 0.057399, 0.08686, and 0.112841 (R(1-15) dataset).

We also compared the KPG results with the ground truth (generating curves). The mean Hausdorff distances KPG yields are 0.2069 ± 0.0064 (*Spiral* dataset), 0.1571 ± 0.0515 (*Helix* dataset), and 0.4145 ± 0.20 (*Rune* dataset). These are considerably higher than the distances our method yields from the ground truth. We did not compute these distances for KDE-SCMS, since this method does not deliver a *proper* PG, *i.e.*, a set of curves, but just a point density. Computing the distance to the ground truth would require a way to simplify and connect this point cloud, which is far from trivial. Overall, we conclude that our method delivers very close results to the ground truth curves. This is, to our knowledge, the first time that the quality of PGs has been quantified explicitly by distance measures.

Parameter setting: The parameters of our method (see Alg. 1) are set as follows: We use the defaults $\tau_{SPG} = 0.05\|\partial\Omega\|$ and $g_{SPG} = 1$, as discussed in Sec. 3.4. The image resolution r_{SPG} is set as a function of the visual space available to depict the final summarizations. If we want to show a large SPLOM-like matrix of PGs, then $r_{SPG} = 256$ suffices. For examining individual summarizations, $r_{SPG} = 512$ gives good results. Larger resolutions can be easily and still quickly computed (see Sec. 4.3), but are not needed, since our PGs are intented to show *summarizations*.

The remaining parameters to discuss are m_{SPG} and σ_{SPG}. As stated in Sec. 3.4 larger m_{SPG} values yield coarser PGs. To analyze this, we generate 10K samples with the *Helix* generator for various image resolutions r_{SPG} (datasets in the supplementary material). Figure 9 shows that the distance D to ground truth (Eqn. 5) increases with m_{SPG} and that the increase is faster for lower-resolution images. Using the default setting $m_{SPG} = 0.05 \cdot r_{SPG}$ yields distances D of roughly 0.02, which we consider acceptable in practice.

To examine the effect of parameter σ_{SPG}, we consider the *Helix* datasets at $r_{SPG} = 256^2$ resolution for varying kernel sizes. Figure 10

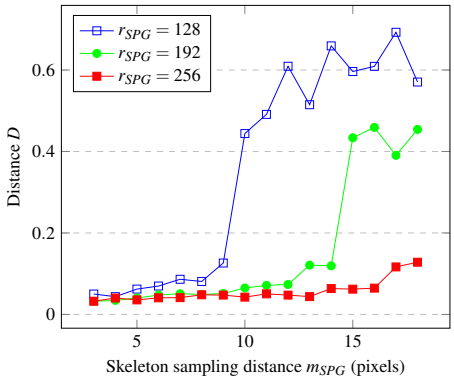


Figure 9: Distance between principal graphs and generating curves for different sampling distances m_{SPG}.

shows that the distance D between our results and ground truth is overall low up to roughly $\sigma_{SPG} = 5$. Also, D increases faster for noisier datasets. Hence, a good default setting is $\sigma_{SPG} = 5$.

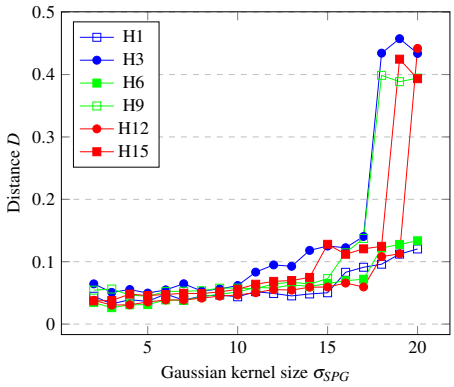


Figure 10: Distance between principal graphs and generating curves for different datasets and values of the kernel size σ_{SPG}

Density effect: Summarizing purely a scatterplot's shape (e.g., [28]) can be misleading, as the point density is not captured. Figure 11 shows this for point distributions within the shape for a linear function $y = x$ with Gaussian noise added in the y dimension. We first added points equally above and below the line $y = x$ for a given noise amount, generating a non-skewed distribution. We next generated two skewed versions – a positively-skewed one with 3 to 1 points above *vs* below the line, and a negative-skewed distribution, with a 1 to 3 points ratio, respectively. Figure 11(top) shows density plots of these three scatterplots, with higher densities mapped to brighter colors, and the generating line in green. Since the overall shapes of the scatterplots are the same (only their 'internal' densities vary), methods that purely capture the scatterplot *shape* [28] would summarize all three such plots identically. In contrast, our method correctly shifts the PG below (Fig. 11e, red line), respectively above (Fig. 11f, red line) the generating line (green). The Hausdorff distances are 0.02476 (non-skewed to positively skewed distributions), 0.0284 (non-skewed to negatively skewed distributions), and 0.03592 (positively to negatively skewed distributions). As expected, the difference between the two skewed distributions is larger than the differences from non-skewed to any skewed distribution.

Finally, let us compare how *scagnostics metrics* would differentiate between the scatteplots in Fig. 11. Graph-theoretic scagnostics [49] generate an MST from samples binned according to the overall number of points within each bin. If this count falls under

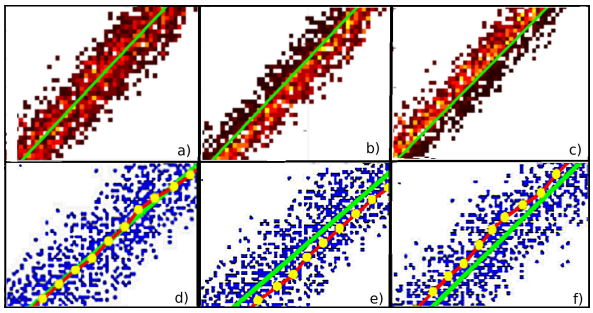


Figure 11: (top) Differently skewed data distributions for scatterplots of same shape: a) unskewed; b) negatively skewed; c) positively skewed. [28] computes equal curves regardless of variation in density. (bottom) SPG captures the skewed distributions leading to respectively shifted PGs (red) , (e) below the curve for negatively skewed distribution and (f) above for positively skewed, (d) overlapping the generating curve (green).

Measure	Non-skewed	Skewed+	Skewed-
Outlying	0.0453	0.0456	0.0455
Skewed	0.5980	0.6020	0.5993
Clumpy	0.0076	0.0089	0.0119
Sparse	0.0140	0.0141	0.0141
Striated	0.0213	0.0174	0.0213
Convex	0.4920	0.4928	0.4928
Skinny	0.5798	0.5783	0.5783
Stringy	0.2753	0.3056	0.2858
Monotonic	0.9793	0.9852	0.9849

Table 2: Graph-theoretic scagnostics measures (computed using R [48]) for differently skewed datasets of same shape deliver (almost) the same result.

a given threshold, the bin size is reduced. Given an equal number of points in S, the point count per bin does not diverge greatly, so the derived MST is similar. Table 2 shows the metrics for the nine features computed by the R "scagnostics" package [48]. These show that the three scatterplots in Fig. 11 are found quite similar – which, as we argued, they are not, density-wise. In particular, the "Skewed" metric characterizes the non-skewed plot as equally skewed as the positively and negatively skewed plots. Dang and Wilkinson define a distance in scagnostics space as the squared Euclidean distance in the feature space [9]. According to this metric, the distances are 0.00098879 (non-skewed to positively-skewed), 0.00016473 (non-skewed to negatively skewed), and 0.00042364 (positively skewed to negatively skewed), thus rating the differences between the three scatterplots in Fig. 11 as smaller than 0.001, i.e., considering them practically identical even though obviously having different skews, which is not desirable. Separately, skeleton-based scagnostics [28] generates identical summarizations (skeleton curves) from all three plots, as already mentioned above. Hence, any metrics derived from it would then find the three scatterplots identical. As already outlined, our method readily distinguishes these three plots as being different.

5 Visual Multidimensional Data Analysis

Besides efficiently solving the computational problem of extracting principal curves for arbitrary-shape-and-density scatterplots, our PGs can facilitate multidimensional data analysis by providing summarizations. Moreover, we can use our proposed distance (Eqn. 5) to compute similarites between multiple PGs summarizing scatterplots obtained by different methods, as follows.

When using dimension selection, the set of all scatterplots within a SPLOM can be compared as proposed by e.g. Matute et al. [28]. Alternatively, one can sample the space of all linear projections from

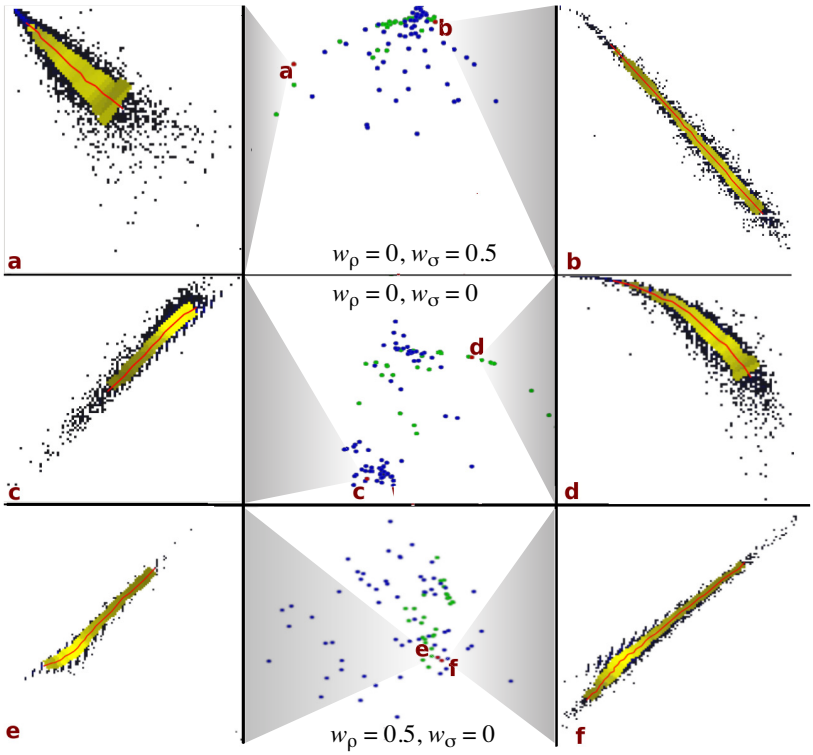


Figure 12: Visual data analysis of *abalone* dataset [11]. Middle column: MDS plots of 100 scatterplots for different weights w_σ and w_ρ of the extended Hausdorff distance. Left and right columns: Selected scatterplots from the MDS plot in the respective row. Top row: decrease of variation from left (a) to right (b) in the MDS plot ($w_\sigma = 0.5$). Middle row: Standard Hausdorff distance shows linear structures (c) at the bottom of the MDS plot and curved structures (d) at the top. Bottom row: Elongated scatterplots with same density behaviour along the PC show high similarity in MDS plot ($w_\rho = 0.5$).

a given d-dimensional space to a 2D visual space using some sampling strategy. SPLOM views are a particular case hereof. Having computed the pairwise distances of all considered scatterplots, the resulting distance matrix can be fed to an MP method such as multidimensional scaling (MDS) to show the similarities of all considered scatterplots.

Since we augmented our PG summarizations by local properties (local density and deviation, Sec. 3.5), we extend our distance D to account for these properties. For this, we replace DT_G in Eqn. 5 by

$$DT'_G(\mathbf{x},Q) = (1-w_\rho-w_\sigma)DT_G(\mathbf{x}) + w_\rho(\rho_{FT_G(\mathbf{x})} - \rho_{FT_Q(\mathbf{x})})^2 + w_\sigma(\sigma_{FT_G(\mathbf{x})} - \sigma_{FT_Q(\mathbf{x})})^2,$$

i.e., a weighted sum of shape distances captured by the PG, local densities ρ, and deviations σ. The weights w_ρ and w_σ can be chosen by the user during an interactive exploration session to give the respective terms more or less influence. Note that, when comparing random linear projections, the orientation of the plots shall not be captured by the (extended) distance. Rotational and translation invariance can be achieved by performing rotation and alignment transformations on the PGs automatically. We apply a simple method that aligns G and G' by translating G' to G's origin. Rotations of G' around the origin are created by sampling the 360° rotation space. The minimal (extended) distance obtained from all these transformations can then be used as a suitable distance measure.

We apply this visual analysis approach to the real-world multidimensional *abalone* dataset from the UCI Machine Learning Repository [11] (4,177 points in 8 dimensions). The sampling strategy uses all SPLOM views plus 72 randomly added linear MPs, i.e., 100 scatterplots in total. Figure 12 shows the resulting MDS plots for different weights w_ρ and w_σ. SPLOM views are shown by green points in the MDS view; other linear projections are shown as blue points. Individual points can be selected to see the respective scatterplot summarizations using our augmented PGs. The center image in Fig. 12 shows the MDS using the 'raw' distance D (Eqn. 5). We see a roughly linear distribution. Plots at the bottom (Fig. 12c) are summarized as a single, quite straight, curve. PG's of plots at the top (Fig. 12d) show an arching behavior. If we increase the weight w_σ of the standard deviation, the MDS plot (Fig. 12 (center, top)) shows scatterplots with thin elongated structures to the right (Fig. 12a) and scatterplots with more spread to the left (Fig. 12b). For increased density weights w_ρ, we show two PGs of scatterplots that are close in the MDS plot (Fig. 12 (center, bottom)). We see that the two scatterplots (Fig. 12e,f) indeed have similar shapes and tailed densities.

6 Conclusion

We have proposed a fast, simple to use, and robust method for computing principal graphs (PGs) from scatterplots. We take advantage of the relationship between medial axes of a compact shape representation of a scatterplot and PCs maintaining the self-consistency criterion. Our approach outperforms previous PG algorithms by being on average two orders of magnitude faster; can treat any 2D scatterplot geometry and/or topology; behaves robustly in the presence of noise; and yields results that are closer to the ground truth (data-generating curves) than competing methods. Additionally, we show how our PGs can compactly summarize local point-distribution characteristics (density and standard deviation). We define an extended Hausdorff distance between such augmented PGs and show how it can be applied for interactive multdimensional data analysis.

Several future work directions are possible: First, given the very

low computational cost, our method could be used to summarize higher-dimensional data than 2D scatterplots, in an interactive exploration way. More refined local properties besides density and standard deviation can be computed and depicted on the PGs, leading to more powerful and insightful summarizations of data distributions. We can apply the fast skeletonization implementation [13] to our method to explore interactively various PG summarizations of scatterplots of hundreds of thousands of points by changing the PG parameters in real time. Finally, now that we have a fast, generic, and robust PG algorithm, we can study how the produced summarizations can support more involved visual analytics tasks including large and complex scatterplots.

Acknowledgments: This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under contract LI-23/1.

REFERENCES

[1] J. D. Banfield and A. E. Raftery. Ice floe identification in satellite images using mathematical morphology and clustering about principal curves. *J Am Stat Assoc*, 87(417):7–16, 1992.

[2] M. Behrisch, M. Blumenschein, N. W. Kim, L. Shao, M. El-Assady, J. Fuchs, D. Seebacher, A. Diehl, U. Brandes, H. Pfister, et al. Quality metrics for information visualization. In *Comput Graph Forum*, vol. 37, pp. 625–662, 2018.

[3] C. M. Bishop et al. *Neural networks for pattern recognition*. Oxford Univ. Press, 1995.

[4] K.-Y. Chang and J. Ghosh. Principal curves for nonlinear feature extraction and classification. In *Applications of artificial neural networks in image processing III*, vol. 3307, pp. 120–130. Intl. Society for Optics and Photonics, 1998.

[5] K.-Y. Chang and J. Ghosh. A unified model for probabilistic principal surfaces. *IEEE TPAMI*, 23(1):22–41, 2001.

[6] W. S. Cleveland and R. McGill. The many faces of a scatterplot. *J Am Stat Assoc*, 79(388):807–822, 1984.

[7] L. Costa and R. Cesar. *Shape Analysis and Classification*. CRC Press, 2001.

[8] R. da Silva, P. Rauber, R. Martins, R. Minghim, and A. Telea. Attribute-based visual explanation of multidimensional projections. In *Proc. EuroVA*, 2015.

[9] T. N. Dang and L. Wilkinson. Scagexplorer: Exploring scatterplots by their scagnostics. In *Proc. IEEE PacificVis*, pp. 73–80, 2014.

[10] P. F. Delicado Useros. Another look at principal curves and surfaces. *J Multivariate Anal*, 77(1):84–116, 2001.

[11] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017. `http://archive.ics.uci.edu/ml`.

[12] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM review*, 41(4):637–676, 1999.

[13] 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.

[14] A. Falcão, J. Stolfi, and R. Lotufo. The image foresting transform: theory, algorithms, and applications. *IEEE TPAMI*, 26(1):19–29, 2004.

[15] M. Friendly and D. Denis. The early origins and development of the scatterplot. *J Hist Behav Sci*, 41(2):103–130, 2005.

[16] S. Gerber, P.-T. Bremer, V. Pascucci, and R. Whitaker. Visual exploration of high dimensional scalar functions. *IEEE TVCG*, 16(6):1271–1280, 2010.

[17] P. Getreuer. A survey of Gaussian convolution algorithms. *Image Processing Online*, 2013:286–310, 2013.

[18] T. Hastie and W. Stuetzle. Principal curves. *J Am Stat Assoc*, 84(406):502–516, 1989.

[19] J. Henrikson. Completeness and total boundedness of the hausdorff metric. *MIT Undergrad J Math*, 1:69–80, 1999.

[20] W. Hesselink and J. Roerdink. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. *IEEE TPAMI*, 30(12):2204–2217, 2008.

[21] K. E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized voronoi diagrams using graphics hardware. In *Proc. ACM SIGGRAPH*, pp. 277–286, 1999.

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

[23] B. Kégl and A. Krzyzak. Piecewise linear skeletonization using principal curves. *IEEE TPAMI*, 24(1):59–74, 2002.

[24] B. Kégl, A. Krzyzak, T. Linder, and K. Zeger. A polygonal line algorithm for constructing principal curves. In *Adv Neur In*, pp. 501–507, 1999.

[25] K. Lang. Learning to tell two spiral apart. In *Proc. Connectionist Models Summer School*, pp. 52–59, 1989.

[26] S. Lloyd. Least squares quantization in pcm. *IEEE T Inform Theory*, 28(2):129–137, 1982.

[27] R. Martins, D. Coimbra, R. Minghim, and A. Telea. Visual analysis of dimensionality reduction quality for parameterized projections. *Computers & Graphics*, 41:26–42, 2014.

[28] J. Matute, A. C. Telea, and L. Linsen. Skeleton-based scagnostics. *IEEE TVCG*, (1):1–1, 2018.

[29] M. Mirzargar, R. T. Whitaker, and R. M. Kirby. Curve boxplot: Generalization of boxplot for ensembles of curves. *IEEE TVCG*, 20(12):2654–2663, 2014.

[30] U. Özertem and D. Erdogmus. Principal graphs and piecewise linear subspace constrained mean-shift. In *Machine Learning for Signal Processing*, pp. 438–443. IEEE, 2008.

[31] U. Özertem and D. Erdogmus. Locally defined principal curves and surfaces. *J Mach Learn Res*, 12(Apr):1249–1286, 2011.

[32] A. V. Pandey, J. Krause, C. Felix, J. Boy, and E. Bertini. Towards understanding human similarity perception in the analysis of large sets of scatter plots. In *Proc. ACM CHI*, pp. 3659–3669, 2016.

[33] F. V. Paulovich, F. Toledo, G. P. Telles, R. Minghim, and L. G. Nonato. Semantic wordification of document collections. *Computer Graphics Forum*, 31(3):1145–1153, 2012.

[34] C. K. Reddy, S. Pokharkar, and T. K. Ho. Generating hypotheses of trends in high-dimensional data skeletons. In *Proc. IEEE VAST*, pp. 139–146, 2008.

[35] A. Sarikaya and M. Gleicher. Scatterplots: Tasks, data, and designs. *IEEE TVCG*, (1):402–412, 2018.

[36] T. Schreck, T. von Landesberger, and S. Bremm. Techniques for precision-based visual analysis of projected data. *Inf. Vis.*, 9(3):181–193, 2010.

[37] K. Siddiqi and S. Pizer. *Medial Representations: Mathematics, Algorithms and Applications*. Springer, 1999.

[38] C. Sorzano, J. Vargas, and A. Pascual-Montano. A survey of dimensionality reduction techniques, 2014. *arxiv.org/pdf/1403.2877*.

[39] Y. Sun and M. G. Genton. Functional boxplots. *J Comput Graph Stat*, 20(2):316–334, 2011.

[40] A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, and A. Telea. 3D skeletons: A state-of-the-art report. *Comput Graph Forum*, 35(2):573–597, 2016.

[41] A. Telea. Feature preserving smoothing of shapes using saliency skeletons. In *Proc. VMLS*, pp. 153–170. Springer, 2012.

[42] A. Telea and O. Ersoy. Image-based edge bundles: simplified visualization of large graphs. *Comput. Graph. Forum*, 29(3):843–852, 2010.

[43] A. Telea and J. J. van Wijk. An augmented fast marching method for computing skeletons and centerlines. In *Proc. VisSym*, pp. 251–259. Springer, 2002.

[44] R. Tibshirani. Principal curves revisited. *Statistics and computing*, 2(4):183–190, 1992.

[45] J. W. Tukey and P. A. Tukey. Computer graphics and exploratory data analysis: An introduction. *The Collected Works of John W. Tukey: Graphics: 1965-1985*, 5:419, 1988.

[46] M. van der Zwan, V. Codreanu, and A. Telea. CUBu: Universal real-time bundling for large graphs. *IEEE TVCG*, 22(12):2550–2563, 2016.

[47] J. J. Verbeek, N. Vlassis, and B. Kröse. A k-segments algorithm for finding principal curves. *Pattern Recogn Lett*, 23(8):1009–1017, 2002.

[48] L. Wilkinson and A. Anand. scagnostics: Compute scagnostics-scatterplot diagnostics, 2012. *R package version 0.2-4*.

[49] L. Wilkinson, A. Anand, and R. Grossman. High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions. *IEEE TVCG*, 12(6):1363–1372, 2006.

[50] A. Yates, A. Webb, M. Sharpnack, H. Chamberlin, K. Huang, and R. Machiraju. Visualizing multidimensional data with glyph sploms. *Comput Graph Forum*, 33(3):301–310, 2014.