Ways of Visualizing Data on Curves

Benjamin Bach bbach@inf.ed.ac.uk
University of Edinburgh, Edinburgh, UK

Qiuyuan Ren qiuyuanren1001@gmail.com
University of Edinburgh, Edinburgh, UK

Charles Perin charles.perin@free.fr
City University, London, UK

Pierre Dragicevic pierre.dragicevic@inria.fr
Inria, Saclay, France

Visual variables used to encode data on the curve in the enrichment stage. Image: Qiuyuan Ren.
Ways of Visualizing Data with Curves

Abstract
This paper reviews the many ways curves are used to encode data in information visualization. As part of our review, we introduce a curve-based visualization framework where data can be encoded in two major ways: i) through a curve's shape (a process we call embedding) and ii) through a curve's local visual attributes (a process we call enrichment). Our framework helps describing and organizing the rich design space of curve-based data visualizations, and offer inspiration for novel data visualizations.

Author Keywords
Information visualization, geographic visualization, temporal visualization, curves.

Introduction
For centuries, artists and scientists have exploited and commented on the rich expressive power offered by curves. For Kandinsky [17], a curve results from a point moving on a paper's surface; at any given point in time, the point can change its direction, speed, and appearance, thereby influencing the shape and final appearance of the curve. Moving from paper and canvases to computer-generated graphics opened up an even wider array of possibilities, creating a vast design space to tap into for communicating emotions, ideas, and facts. In this article, we focus on the use of computer-generated curves for communicating data.
Lines and other types of curves are pervasive in statistical charts and in information visualization (InfoVis), especially when visualizing temporal data [8]. Curves form the basis of elementary visualizations such as line charts [33], trajectory visualizations [2, 35], and connected scatterplots [15]. Meanwhile, new curve-based visualizations are routinely proposed to convey information such as network relations [3], patterns of narration in movies [18], or the sonic topology of poems [22].

Despite the pervasive use of curves, InfoVis researchers have only started to chart and study their design space in a systematic fashion [16, 25]. There is still a lack of general overviews of the many ways curves have been used to (and can potentially be used to) encode data.

For Jacques Bertin [6], there are three major ways to visually represent data elements: through points, lines (i.e., curves), or areas. These are called visual marks. Once visual marks have been decided, visual variables such as size, luminance or texture can be used to encode data attributes. Some pairings between visual variables and visual marks are more sensible than others. For example, texture is best applied to areas, while size is best applied to points. Curves can accommodate a rich set of visual variables, and some visual variables have been specifically created for curves. These include beginning and ending symbols (e.g. arrow-heads), particles [25, 18], and parallel lines [18]. Curves also support many creative variations of classical visual encodings which can be either applied to the whole curve or to specific curve segments. As a result, there is a large design space for encoding data through curves.

In this article, we review how curves are used to convey information in InfoVis. Our review offers a catalog of prominent examples of information effectively communicated through curves, as well as a structured description of the different ways curves can be used to encode data. After introducing the general outline of our framework, we discuss how information can be conveyed through a curve’s shape (embedding), and then discuss how information can be conveyed through a curve’s local visual attributes (enrichment).

**Methodology**

To derive our framework, we collected visualizations and infographics that convey data through curves. Our search was initially guided by our general knowledge of InfoVis stemming from years of attending major visualization conferences, as well as teaching InfoVis classes to Master students. Additional resources we employed include Best American Infographics [12], Graphics Semiology [6], Dear Data [21], Information is Beautiful Awards, Visual Complexity [20], Brehmer’s et al. survey on timelines [8], the Space-time cube survey [2], Pinterest and general Google Image search.

Our collection features classical visualizations such as line charts, node-link diagrams, and trajectory visualizations, as well as less common designs such as connected scatterplots [15] and time curves [4]. In an attempt to identify visual encodings that are unknown or underused in InfoVis, we also collected examples of curve depictions from graphic design and art.

In total, we collected over 60 images of visualizations, infographics, and artwork. This number being too large to fit a single article, we only present a selection of prominent and representative examples. An extensive list of our examples can be found online: [http://dataoncurves.wordpress.com](http://dataoncurves.wordpress.com).
Framework for Curve Visualizations
For simplicity, we focus on single-curve visualizations although our framework easily expands to capture multi-curve visualizations. In such visualizations, data can be encoded in two major ways: i) using spatial embedding (i.e., through the curve’s shape); and ii) using enrichment (i.e., enrich the style through the curve’s visual attributes). The two methods can be combined.

Figure 1 illustrates the general process of encoding data with a curve. This process is data-agnostic as long as the dataset consists of an ordered list of data elements. A data element refers to an entity in the dataset (e.g., a person or a location), defined along an arbitrary number of attributes (e.g., age, position, time). Data elements are typically stored as rows in a data table, while attributes are stored as columns. Data elements need to be ordered. Their ordering can be either a) intrinsic (e.g., defined by the order of rows in a data table), b) based on values of an attribute (e.g., timestamp), or c) derived from a variety of attributes (e.g., minimization of pairwise distances between data elements [5]).

We define the curve encoding pipeline illustrated in Figure 1 as the following three-stage process:

1. In the embedding stage, each data element is mapped to a point with a position in space. Positions can be calculated through a variety of algorithms and visualization techniques, which will be discussed in the next section.
2. In the connection stage, the points are joined in order to produce a continuous curve. A curve segment is added between each pair of consecutive points, according to the ordering of the data elements. Although the curve segments can be straight, other interpolation techniques can be used (Bezier, splines, etc.). Taken together, we refer to the embedding + connection stages as curve embedding.
3. In the enrichment stage, data attributes are mapped to local visual attributes of the curve, such as thickness, texture, or color. Since there is a wide design space for enrichment, much of this article will focus on this stage.

We now describe the embedding stage in more detail. We will then turn to the enrichment stage.

**Embedding:** Encoding Data with a Curve’s Shape
Curve embedding provides rich opportunities for conveying data visually. A curve can take on many forms varying in complexity, from straight lines to smooth curves to zig-zag patterns. Global and local geometric features of a curve (e.g., sharp turns, loops, crossings) immediately stand out, and can tell captivating stories about the data [4, 15].

Existing curve embedding techniques can be categorized according to i) the dimensionality of the encoded data and ii) the dimensionality of the curve itself. Data dimensionality defines how many data attributes dictate the curve’s
shape, i.e., how many of these attributes are used to position the points in the embedding stage of Figure 1. It can vary from zero to an arbitrary number of dimensions. In contrast, curve dimensionality refers to the dimensionality of the resulting curve. It is generally either 1D (i.e., a straight line), 2D (i.e., a planar curve), or 3D.

Here is an overview of curve embedding approaches classified according to their data dimensionality:

**0D**—In zero-dimensional approaches, the shape of the curve is not dictated by data but is decided in advance and fixed. Examples include timelines whose layout is data-independent and where data is encoded through the enriching process exclusively. In terms of curve dimensionality, while many timelines are 1D (i.e., straight lines [26, 8]), timelines can also take on complex shapes on the 2D plane (spirals, boustrophedons, etc. [8]), and even possibly in 3D space.

**1D**—Several curve embedding techniques exist that encode a single data attribute. Some line charts fall in this category, when the visualized dataset has no explicit time attribute (e.g., in time-series data where time intervals are constant, or in event sequences without time stamps). In such line charts, points are typically evenly spaced out on the horizontal axis, while their vertical position is decided by a single data attribute. Other examples include curves whose layout is fixed in advance but where points are positioned along the curve depending on some data attribute (e.g., time). Finally, a less common technique is the turtle-walk, where curves are constructed by adding each new data point at a fixed distance from the previous one, and at an angle that depends on the value of some data attribute. This technique has been used to visualize digits of π [31] but can also be used with continuous quantitative attributes.

**2D**—Many classical curve-based visualization techniques map two data attributes onto the two dimensions of the plane. These include all line charts that encode an explicit time attribute (typically on the x-axis), plus another data attribute (typically on the y-axis). A related technique is the connected scatterplot, where one data attribute is mapped to x while another one is mapped to y [15]. In contrast to the line chart, the data attribute mapped to the x-axis does not need to be monotonically increasing. While connected scatterplots can be used to encode abstract data, they can also be used to encode spatial data, in which case they simply become 2D trajectory visualizations. While such Cartesian mappings are the most common, non-Cartesian mappings of two data attributes onto the 2D plane are also possible. For example, one could draw a line chart in polar coordinates, or use a modified turtlewalk technique where both point spacing and point angle encode data.

**3D**—Some curve-based visualization techniques use three data attributes in the embedding process. A common approach is the space-time cube curve, where two data attributes are mapped to x and y respectively, while a time attribute is mapped to z [2] (Figure 2(a)). This technique can be seen as an extension of line charts to two non-temporal attributes, and it is especially useful for visualizing two-dimensional trajectories [19]. Connected scatterplots and trajectory visualizations can also be generalized to three dimensions. All such visualizations use 3D embeddings, meaning that the produced curves are generally not planar. Such curves can be either be displayed without any loss of information through manual or digital fabrication1, or they can be projected on a 2D medium [4].

---

1For two examples, see dataphys.org/list/wire-models-of-factory-worker-movements/ and dataphys.org/list/physical-space-time-cubes/
nD—Through the use of dimension reduction techniques [14],
it is possible to produce 2D or 3D curves based on an arbi-
trarily large number of data dimensions. Although dimen-
sion reduction techniques have been mostly used to pro-
duce point-based visualizations (e.g., scatterplots), the
time curve approach [4, 30] (Figure 2(b)) illustrates how
multidimensional scaling (MDS) can be used to produce
curves that encode an arbitrarily large number of dimen-
sions, while remaining relatively easy to interpret even for
a non-technical audience. While such techniques can be
powerful, they necessarily discard information in the data.

This overview helps classify curve-based visualizations ac-
cording to the data and the encoding that are used to cre-
ate the basic shape of the curve. In the next section, we
present the last stage of the curve encoding pipeline shown
in Figure 3: curve enrichment.

**ENRICHMENT: Encoding Data with a Curve’s Attributes**
Enrichment is a stage where additional data attributes are
mapped to visual attributes of the curve. In other words, it is

The visual variables covered in this section are illustrated in
Figure 3. **Thickness** refers to the width of the curve’s stroke (Fig-
ure 3(a)). It is one of the most common visual variables
used to enrich curves. For example, thickness is commonly
used to represent the strength of traffic flows, such as the
Figure 4: Examples for the use of thickness and color: a) strength of traffic (train passengers) [24] (© L.Nguyen), b) transported units (animals) by Minard, (© C.J.Minard), c) Baby names (©N.Bremer), d) uncertainty in line charts [27] (© R.Ravindrarajah).

number of cars or people traveling through a road per unit of time [24, 11] (Figures 4(a) and 4(b)). Mapping traffic flow to curve thickness is an old practice that dates back from at least Charles Joseph Minard, with his Napoleon’s March [23] (Figure 5(a)) and other flow maps.² GIS³ maps line thickness to road throughput such that traffic jams become visible as narrow curve segments, while fluent traffic appears as thick segments. Finally, most road maps and street atlases use curve thickness to convey road importance.

Curve thickness has also been used to convey temporal information—for example, time and speed in trajectory visualizations [25], relative time intervals in time curves [4], and absolute time in connected scatterplot visualizations⁴ (Figure 8(a)). Other usages include the communication of link strength in graphs [7], elevation in trail maps [34] (Figure 5(b)), variance in line charts [27] (Figure 4(d)), and highest position in ranking charts [9] (Figure 4(c)).

Hue is the ‘type’ of color that is used to color a curve and is often referred to simply as ‘color’. This section focuses on hue only; we describe the other components of ‘color’ (transparency, saturation, and brightness) in the next subsection. Hue is commonly used to represent different objects, and to help visual grouping of curves of the same type. Examples include types of traffic flows (Figure 4(b)), type of roads.

Other than for grouping, hue has been to encode data values at specific positions along the curve, for example, to show intensity of traffic flows and related speed of moving objects [11, 10]. An interesting use of hue is found in Minard’s work Napoleon’s march (Figure 5(a)) which uses hue to indicate both time of travel and direction; orange curve segments indicate the first part of Napoleon’s campaign with the army marching from west (left) to east (right). The army’s retreat in the opposite direction is colored gray.

²https://sandrarendgen.wordpress.com/2013/06/22/the-forgotten-maps-of-minard/
⁴http://truth-and-beauty.net/projects/remixing-rosling
Figure 5: Examples of thickness encoding on maps: a) Napoleon's march (© C-J. Minard), b) Bike elevation map (© J. Wood).

Figure 6: Examples for the use of hue in linecharts (MS Excel) (© J. Peltier): a) Upwards, downwards, and stable periods, b) value intervals on y-axis (“conditional cart”).

In line charts, hue has been used to encode upwards, downwards, and stable periods (Figure 6(a)) as well as value intervals (Figure 6(b)). In these examples, hue is used redundantly to curve direction (up, down, unchanged) and position of the curve segment on the y-axis respectively. In other designs with a 2D embedding, hue has been used to indicate bivariate orientation: left/right (Figure 7(a)) as well as angles (Figure 7(b)).

Eventually, hue has been used with connected scatterplots (Figure 8) to differentiate between elements and their tra-

Figure 7: Examples for the use of hue to show direction in 2D embeddings: a) Notabilia: Visualizing Deletion Discussions on Wikipedia. Each thread represents an article; green segments lean left and signify acceptance of a change, red segments lean right and mean rejection (© M. Stefaner), b) Exploring the Art hidden in Pi. Each digit (0-9) is encoded through both color and angle (direction). Visualizing the array of digits in the number Pi results in that fractal curve (© N. Bremer).

Figure 8: Examples for the use of hue to show direction in 2D embeddings: a) Notabilia: Visualizing Deletion Discussions on Wikipedia. Each thread represents an article; green segments lean left and signify acceptance of a change, red segments lean right and mean rejection (© M. Stefaner), b) Exploring the Art hidden in Pi. Each digit (0-9) is encoded through both color and angle (direction). Visualizing the array of digits in the number Pi results in that fractal curve (© N. Bremer).


http://notabilia.net
Figure 8: Examples for encoding time and reading direction in connected scatterplots: a) Remixing Rosling (© M.Stefaner), b) Inequality and GDP (© Epoca Magazine, Brazil).

Transparency, saturation, and brightness mean the intensity of the curve and its transparency. As transparency and saturation are often used to achieve this same affect, here we review examples of both, though technically they remain two independent visual variables. Transparency is frequently used to discriminate overlapping or clustered objects (e.g., in Figure 4(c)), yet less to directly encode data. One example that uses transparency to encode data is Robertson et al. [28] who use transparency to encode time in connected scatterplots, with more transparent point signaling points further in the past. This mimics the effect of fading out. A similar effect is achieve in time curves [4] (Figure 2(b)), using a mixture of brightness, saturation, and hue (a color range from lighter shinier to a darker orange) to indicate reading direction (start to end). This is a prominent example because it is one of the very few that encode the reading direction on a curve using a visual attribute rather than text labels such as years.

Texture refers to the local structure of curve. Other than Bertin, we differentiate between particles (e.g., dashed lines and different types of strokes) and textures. Particles are explained in the next subsection and refer to designs where the curve is represented through a set of individual visual elements. Textures, in our sense, are complex surface structures which would require shaders and/or photographs to be mapped onto the curve in both 2D and 3D visualizations of curves.

Bertin [6] offers many examples of textures for curves, e.g., to represent rivers, roads, and other curves on maps (Figure 9(a)). In these examples, the original curve shape is altered through specific renderings such as meanderings, small turns, double lines, pointilism, etc. A similar example from Dear Data [21] (Figure 9(b)) shows example textures—zigzag lines of different frequency and almost curly lines—aside examples for particles (curves consisting of strokes and points) as explained in the next subsection.

A specific instance of texture is text, a technique inspired by calligrams. Text means words and text resembling apped onto the trajectory of the curve and its shape. Text curves are used to show spatial data where text alone forms the graphical features (Figure 9(c)) [1]. Every curve consisted of text representing the road’s name. Beyond curves as regarded in this survey, words have been used in node-link diagrams to depict link types [32].

Particles can be seen as a specific type of texture, e.g., a dashed line. In this article, particles are individual visual marks, laid out along the curve’s trajectory and resulting in a specific texture according to the Gestalt law of Continuation. A particle’s shape can be anything as simple as a dot.
Figure 9: Examples of texture on curves: a) curves on maps showing rivers, river banks, streets, walls (© J.Bertin), b) particles and zig-zag curves in Dear Data (© S.Posavec [21]), c) text depicting roads [1] (© S.Afzal).

(resulting in a dotted line), a stroke segment (resulting in a dashed line), strokes perpendicular to the direction of the curve, symbols (small crosses in Fig 9(b)) and complexer visual elements such as 3D [13] (Figure 10).

Particles can be applied independently from any encoding of an underlying curve, such as using tickmarks on top of an existing trajectory. For example, Figure 11(a) uses tick marks along trajectories to indicate the number of travel days: the space between two tickmarks shows the space traveled in one day [6]. Bertin provides many examples for trajectories on maps where different particle shapes, alterations, and spacing along the curve, differentiate trajectories (Figure 11(b)). Shapes in this example include arrowheads, arrows with tail, and alternations between points and arrows.

Particles have been eventually studied in more detail by Romat et al. [29] for conveying different information on links in networks. Alongside the common visual attributes (size, color, and shape) their framework defines parameters such as particle pattern (a pattern of particles similar to a Morse code), pattern frequency (visual spacing of a pattern), and particle speed. Each of these parameters can be used to encode different data.

Finally, some animations use particles to show data elements traveling along the trajectory. Number, density and position of the particles represent information in the data. Other examples (e.g. [29]) use parameters of a particle system to encode data, and animating particles can be used to indicate direction of flow along a curve [3, 29].

http://ilda.saclay.inria.fr/flownet
https://www.youtube.com/watch?v=v1kfW1d3zsA
Figure 11: Examples for (static) particles along trajectories [6] (© J.Bertin): a) marks indicating days of travel (one mark=1 day), b) particle types indicate different expeditions.

Conclusion
We have started describing the richness of curve designs in information visualization by introducing the combination of curve embedding and curve enrichment to describe existing curve designs. While our framework is preliminary, it helps in organizing and comparing existing curve designs used in information visualization. We anticipate that it can inform the design of novel visualizations that rely on curves to encode information. For example, all of our curve-specific visual variables can be combined and hence be used to independently encode a set of data attributes for each segment of the curve. Eventually, such a systematic review can inform the evaluation of individual or combinations of curve designs for specific applications, tasks, and datasets.

REFERENCES


