



# A Descriptive Framework for Temporal Data Visualizations Based on Generalized Space-Time Cubes

Benjamin Bach, Pierre Dragicevic, Dominique Archambault, Christophe Hurter, Sheelagh Carpendale

## ► To cite this version:

Benjamin Bach, Pierre Dragicevic, Dominique Archambault, Christophe Hurter, Sheelagh Carpendale.  
A Descriptive Framework for Temporal Data Visualizations Based on Generalized Space-Time Cubes.  
Computer Graphics Forum, 2016, 10.1111/cgf.12804 . hal-01303506

**HAL Id: hal-01303506**

**<https://enac.hal.science/hal-01303506>**

Submitted on 29 Nov 2016

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - ShareAlike 4.0 International License

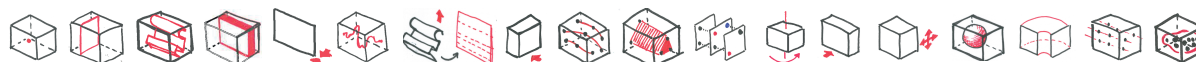
# A Descriptive Framework for Temporal Data Visualizations Based on Generalized Space-Time Cubes

B. Bach<sup>1</sup>, P. Dragicevic<sup>2</sup>, D. Archambault<sup>3</sup>, C. Hurter<sup>4</sup> and S. Carpendale<sup>5</sup>

<sup>1</sup>Microsoft Research—Inria Joint Centre, France, <sup>2</sup>Inria, France;

<sup>3</sup>Swansea University, UK; <sup>4</sup>ENAC, France;

<sup>5</sup>University of Calgary, Canada



**Figure 1:** Examples of space-time cube operations, performed on a generalized space-time cube to describe visualizations.

## Abstract

We present the generalized space-time cube, a descriptive model for visualizations of temporal data. Visualizations are described as operations on the cube, which transform the cube's 3D shape into readable 2D visualizations. Operations include: extracting subparts of the cube, flattening it across space or time, or transforming the cube's geometry and content. We introduce a taxonomy of elementary space-time cube operations and explain how these operations can be combined and parameterized. The generalized space-time cube has two properties: a) it is purely conceptual without the need to be implemented, and b) it applies to all datasets that can be represented in two dimensions plus time (e.g., geo-spatial, videos, networks, multivariate data). The proper choice of space-time cube operations depends on many factors, e.g., density or sparsity of a cube, hence we propose a characterization of structures within space-time cubes, which allows us to discuss strengths and limitations of operations. We finally review interactive systems that support multiple operations, allowing a user to customize his view on the data. With this framework, we hope to facilitate the description, criticism and comparison of temporal data visualizations, as well as encourage the exploration of new techniques and systems. This article is an extension of [BDA\*14].

Categories and Subject Descriptors (according to ACM CCS): H.5.0 [Information Systems]: Information Interfaces and Presentation—General

## 1. Introduction

Temporal datasets are ubiquitous but notoriously hard to visualize, especially rich datasets that involve more than one dimension in addition to time, such as videos, dynamic networks, multivariate and geo-spatial data.

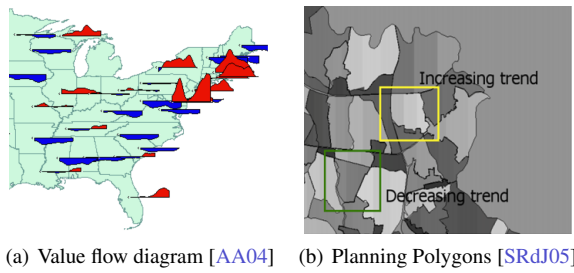
Previous work on novel visualizations for temporal data has dramatically advanced the field of information visualization (Infovis). However, there are so many different techniques today that it has become hard for both researchers and designers to get a clear picture of *i)* what has been done, *ii)* how to compare and chose visualizations, and *iii)* how much of the design space of temporal data visualizations remains to be explored. For similar reasons, teaching this research topic to students is challenging. Therefore, there is a clear need to structure and organize previous work in the area of temporal data visualization.

Part of the problem is that information visualization researchers have mostly focused on *nomenclature*. Most fa-

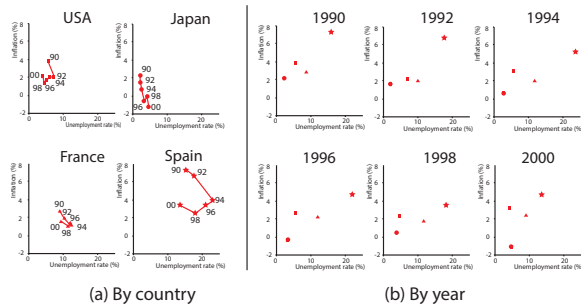
miliar charts have an agreed-upon name, e.g., *small multiples* or *scatter plots*, and this tradition has been continued in infovis, where each newly published visualization technique is given a different name. Many textbooks and surveys list existing techniques by their name, both for general visualizations [Har99] and for temporal visualizations [AMST11].

While names are essential for indexing, retrieval and communication purposes, they are a poor thinking tool. Because there is no convention for naming techniques, names rarely reflect the essential concepts behind a technique. For example, names such as *Value Flow Maps* [AA04] and *Planning Polygons* [SRdJ05] say little about the possible conceptual similarities between the two techniques (see Figure 2). Names can also be ambiguous. For example, the term *small multiples* is commonly used to refer to a specific type of temporal data visualization [Tuf86]. But Figure 3 shows that two visualizations can be based on small multiples despite being different with respect to which aspect of the data they show.





**Figure 2:** Two conceptually similar temporal visualization techniques showing: (a) the evolution of crime statistics per state; (b) the evolution of high school population per district.



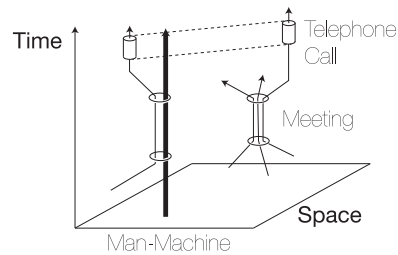
**Figure 3:** Two visualizations using small multiples to show the same indicator data for 4 countries over 6 years, but which are conceptually different.

Recent effort at proposing taxonomies, conceptual models, and design spaces for temporal visualizations mainly categorize visualizations based on analytical tasks or data types (e.g., object movement data [AAH11, AAB\*11, AA12], video data [BCD\*12], or datasets with different temporal and spatial structures [AMM\*07]). While descriptions of visualization techniques by tasks and data types can help choosing a visualization technique for a specific problem, we aim to describe visualizations in their own visualization space, independent from data and task. We contend that it is limitative to reason in a task- and data-agnostic manner, and that this is uncommon in visualization (which emphasizes point designs, point studies and cookbook-type recommendations), but because it is rare enough, it is worth trying.

Our goal is to understand a) how those visualizations relate conceptually, b) to provide a clear terminology, and c) a way of organizing these visualizations that can serve as a basis for prescriptive and evaluative work. In other words, our framework is mostly *descriptive*, but, as we show, it can help better understand the strengths and weaknesses of different visualizations with respect to tasks and data, and better formulate research questions in empirical studies.

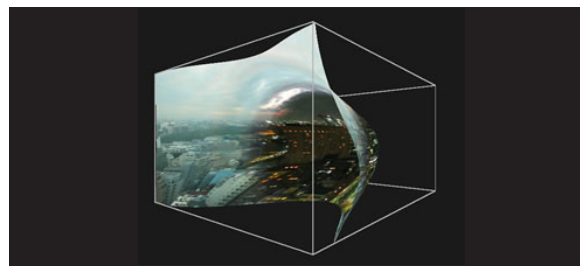
The benefit of a clear and detailed descriptive framework is that it helps connect techniques that are similar and distinguish techniques that are dissimilar. For example, the two techniques from Figure 2 have different names but are the

result of a similar (compound) operation on a *generalized* space-time cube and which we call *repeated drilling*. Figure 3 shows two techniques with the same name (“small multiples”) but resulting from very different space-time cube operations; Figure 3(a) involves operations such as *filtering*, *time flattening* and *space shifting*, while Figure 3(b) is the result of a compound operation we call *time juxtaposing*. By discussing such relations, we hope to improve our understanding of which visualizations are complementary and which ones are possible, but have not yet been realized in visualization. Eventually, we discuss how multiple operations can be combined into an interactive and consistent interface. Our article is targeted towards designers and researchers, in particular students in both domains.



**Figure 4:** A space-time cube based on an illustration by Hägerstrand [H70] in 1970, showing social interactions across space and time.

The term *space-time cube* originates from social geography, where it refers to a 3-dimensional Euclidean space, consisting of a 2-dimensional geographical space plus time. Under the term “time geography”, Torsten Hägerstrand in 1970 described a “*space-time model which could help us to develop a kind of socio economic web model*” [H70, p. 10]. His intention was to analyze people’s behaviour and interactions across space and time (Figure 4); a moving person on a 2D map becomes a static 3D trajectory visualized in a space-time cube. Since then, the space-time cube has been used as a visualization metaphor in a number of interactive visualization systems in geo-visualization [CCT\*99, Kra03], video visualization [FLM00], networks [BPF14b], and art [CI05]. However, it has never been used as a conceptual model for reflecting on visualizing temporal data in general.



**Figure 5:** Khronos projector [CI05] lets users dig into video cubes: here, a scene transitioning from day to night.

With respect to the traditional understanding of a space-time cube—a 3D representation of geographic space plus time—here, we define a *generalized* space-time cube, inspired by Furnas notion of generalized fisheye views [Fur86]. The generalized space-time cube has the following two main properties:

1. A generalized space-time cube is a **conceptual representation** that helps to think about temporal data visualization techniques in general. It does *not* imply an explicit 3D visualization nor a perspective projection thereof. The cube does *not* need to be implemented in the system, used to generate this visualization. For example, the visualizations in Figures 2 and 3 do not show any space-time cube; they are purely 2D visualizations. For most of the examples in this paper, we don't know the exact format of the underlying data (1D, 3D, tables, multivariate, network). However, we imagine a *possible* conceptual cube which the visualization can be derived from. This does not imply how the visualization is implemented or whether the designer was aware of any conceptual cube and generalized space-time cube operations. In Section 6, we discuss systems that explicitly implement the space-time cube and its operations.

2. A generalized space-time cube **does not need to involve geo-spatial data**. Many visualizations (e.g., scatterplots or node-link diagrams) convey abstract, non-spatial data. Those (non-temporal) visualizations all embed data into an abstract and 2D *visualization space*. When data changes over time, such as in GapMinder's animated 2D scatterplots [Ros06], each animation frame can be conceptually thought of as a slice of a space-time cube. In the model of the generalized space-time cube, “space” therefore refers to an abstract 2D substrate that is used to visualize (non-temporal) data at a specific time. We call this space *base plane*. Thus, generalized space-time cubes include traditional geo-spatial space-time cubes, video cubes, matrix cubes, as well as a subset of OLAP cubes as we discuss later.

It is important to stress that *this article is not about space-time cube visualizations*, and that 3D space-time cube representations, like the one in Figure 4, represent only a very small subset of the visualizations we aim to cover. In the remainder of this article, the term *space-time cube* refers to our model, if not indicated otherwise.

Finally, our framework does not consider how space-time cubes are built for a given dataset. A general heuristic is to decide on a 2D visual representation of the data for a given time slice, and then extrude it over time. For example, networks can be shown as node-link diagrams or matrices, multivariate data as scatterplots or barcharts. Such decisions are independent from the cube. We admittedly do not offer a detailed framework that clarifies the different ways concrete datasets can be turned into concrete space time cubes. Instead, we assume that a conceptual 3D space-time cube is already given and focus on how this cube can be transformed to accommodate 2D media like computer displays and pa-

per while remaining legible. We show how such operations are enough to capture most known techniques for visualizing temporal datasets. We mainly consider datasets that involve two dimensions plus time, although we later discuss how our model can be extended to higher dimensionalities. Datasets with a single data dimension plus time may *not* require a space-time cube and can be visualized with timelines and visualizations for time series.

In this paper, we first review common temporal data visualization techniques, and explain how they can be all seen as operations on a generalized space-time cube. We then describe our framework in more detail by providing definitions of key concepts, as well as a taxonomy of elementary space-time cube operations and how they can be combined into compound operations. Then, we explain how choosing effective space-time cube operations can be motivated by the characteristics of the dataset, mostly because different datasets yield different “shapes” or *inner structures* for space-time cubes. We provide a way to characterize such inner structures, based on characteristics of data. We then informally discuss the relative benefits and drawbacks of different space-time cube operations and briefly review empirical evidence from user studies. We finally discuss interactive visual exploration systems that support a range of space-time cube operations, discuss limitations of our framework, and suggest avenues for future work. A catalog of temporal visualizations classified according to our framework can be found at [www.spacetimecubevis.com](http://www.spacetimecubevis.com).

## 2. Static Visualizations as Space-Time Cube Operations

In this section, we illustrate examples of generalized space-time cube operations and how they can be used to describe a range of common *static* visualization techniques for temporal data, all meant for screen or paper media. We focus on a small but representative selection and describe operations informally, often using analogies from photography and art.

While a generalized space-time cube may not be the only way to describe these visualizations, and the designer may not have thought of it in this way, we construct conceptual space-time cubes for our examples, then apply the respective operation(s) that describe the actual visualization. Note that, while in Hägerstrand's original illustration the time axis is vertical, in our illustrations time passes from left to right.

### 2.1. Time Cutting

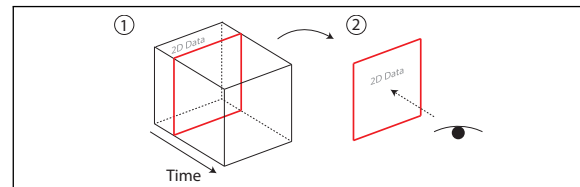


Figure 6: The time cutting operation.

A *time cutting* operation consists in extracting a particular temporal snapshot from the cube to be presented to the viewer. Figure 6 illustrates this operation: the left part (1) shows the initial space-time cube and the temporal snapshot that is being extracted, while the right part (2) shows the resulting image that is presented to the viewer.

For example, consider a photographer who captures a particular instant of a moving scene. If the scene being viewed is represented as a space-time cube (i.e., all possible pictures are piled up to form a cube), then taking a photograph is equivalent to applying a time cutting operation on this cube.

In information visualization, an image produced by time cutting is typically called a *time slice*. But a temporal visualization rarely consists in a single time slice. As we will see in Section 3, time cutting is typically either performed multiple times and used in combination with other operations, or it is used in combination with animation and interaction.

## 2.2. Time Flattening

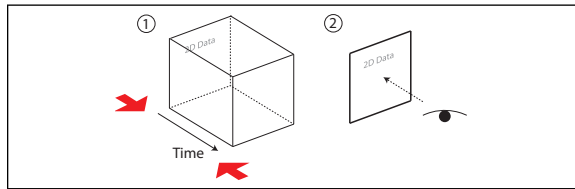


Figure 7: The time flattening operation.

*Time flattening* collapses the space-time cube along its time axis, by merging all time slices into a single 2D image (Figure 7). An analogy is *long exposure photography*, which collapses several seconds, minutes or even hours of a natural scene into a single image.

One of the earliest uses of time flattening is Minard's illustration of Napoleon's march towards Moscow (Figure 8). The illustration shows on a single image the state of Napoleon's army (position, size, key events) at different points in time during the Russian campaign in 1812 [Tuf86]. Another early example is Dr. John Snow's map showing where deaths from cholera occurred in London in 1854 (Figure 9(a)). The map shows events from several days aggregated over time.



Figure 8: A famous example of time flattening: Napoleon's march to Moscow by Joseph Minard [Tuf86].

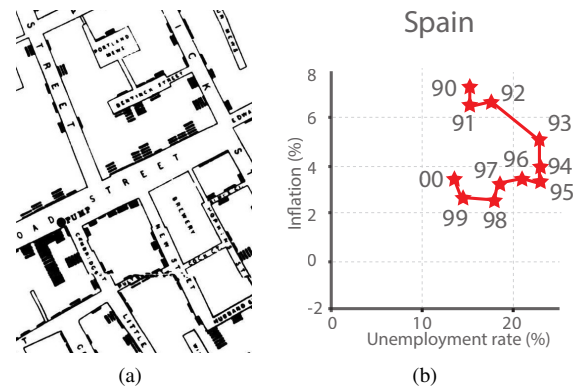


Figure 9: Other examples of time flattening: (a) Detail of the map of the cholera outbreak in London 1854, by Dr. John Snow. Piled bars mark the number of death per house. (b) Connected scatterplot showing the relationship between inflation rate and unemployment in Spain from 1990 to 2000.

Many maps that show temporal data can be seen as time-flattened geo-spatial space-time cubes. But the time flattening technique is not limited to geographical data and has been employed in a large variety of information visualization systems as well as in static data graphics. Figure 9(b) for example, shows the evolution of inflation and unemployment rate in Spain from 1990 to 2000. This diagram can be seen as time-flattened version of a space-time cube created from a scatterplot that contains a single data point evolving over time (*Spain*).

## 2.3. Discrete Time Flattening

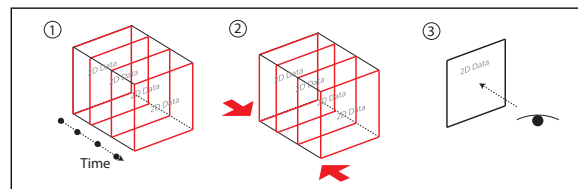
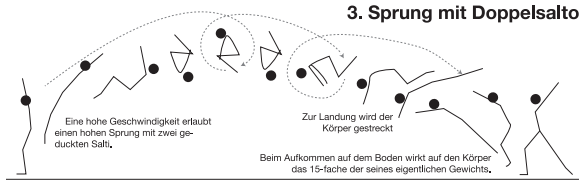


Figure 10: The discrete time flattening operation.

*Discrete time flattening* is similar to time flattening, but instead of merging all time slices, a selection of meaningful time slices is made before combining them (Figure 10).

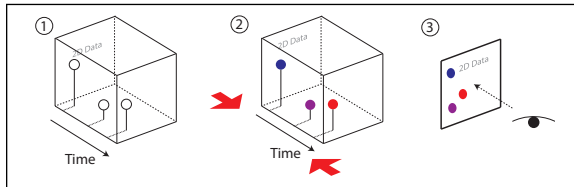
An analogy for discrete time flattening is *multiple exposure photography*, where several photos are taken at different times and blended into a single image. Etienne-Jules Marey pioneered this technique in 1882 with an instrument (the chronophotographic gun) that records 12 photos per second on the same film. He used it to visualize human and animal motion [Mar78]. Modern art has also employed a similar technique to convey movement, e.g., Marcel Duchamp's "Nude Descending a Staircase, No. 2".



**Figure 11:** An example of discrete time flattening. For a better infographic by Megan Jaegerman, see [Tuf].

Tufte [Tuf86] comments on several examples of infographics that employ discrete time flattening. He calls them *sequences*. One of his famous examples is the life cycle of the Japanese beetle [Tuf86]. Figure 11 is a sequence showing a dancer's move. Discrete time flattening has also been used for summarizing videos [BDH04].

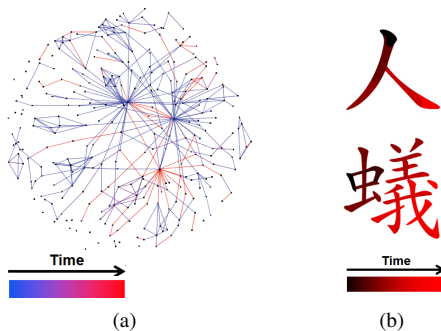
## 2.4. Colored Time Flattening



**Figure 12:** The colored time flattening operation.

The *colored time flattening* operation is similar to the time flattening operation, but time slices are assigned a color before being combined (Figure 12). Although this operation does not map to any photography technique we know of, similar results could in principle be obtained by rapidly switching color filters during a long-exposure photography.

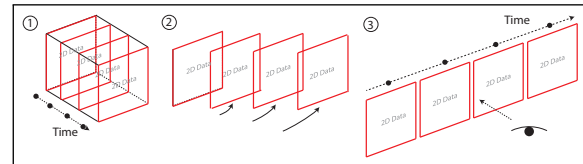
Two examples of visualizations obtained by colored time



**Figure 13:** Two visualizations using colored time flattening. (a) Illustration of a dynamic graph visualization as used in GEVOL [CKN\*03]. (b) Stroke order in Chinese characters [Wik13]; the color legends have been added.

flattening are shown in Figure 13: (a) a dynamic graph where old links (in red) are distinguished from new links (in blue) [CKN\*03]; (b) Chinese characters where first strokes (in black) are distinguished from later strokes (in red) [Wik13]. Minard's map (Figure 8) also makes use of a simplified form of colored time flattening, since the army's forward march and return are distinguished using two different colors.

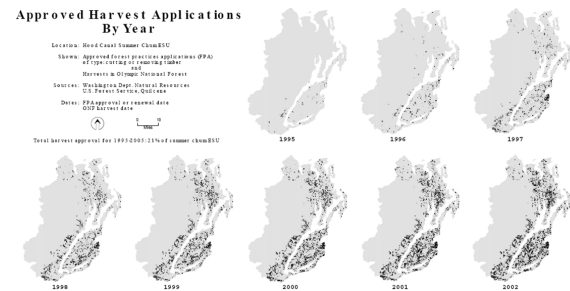
## 2.5. Time Juxtaposing



**Figure 14:** The time juxtaposing operation.

*Time juxtaposing* consists in extracting multiple time slices and placing them side-by-side or on a grid (Figure 14).

An analogy is Eadweard Muybridge's *multiple camera chronophotography* [Muy87]. In contrast with Marey, Muybridge used multiple cameras that recorded snapshots on *different* locations on the film. He used it for the scientific study of for example horse gaits, and his pictures famously settled the question as to whether horses have all four feet off the ground while trotting. Time juxtaposing is also the base for many forms of sequential art, from ancient Egyptian murals and Greek vase paintings to today's comics [McC94, BKH\*16].

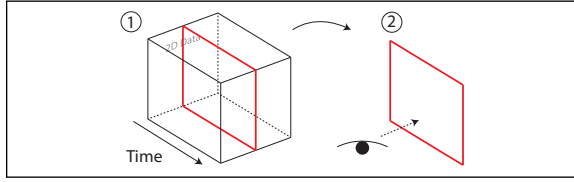


**Figure 15:** Time juxtaposing showing approved forest harvest applications across 10 years [Gre11].

Time juxtaposing is often used in information visualization to show temporal data such as time-evolving maps, trajectories in space [TBC13] and dynamic graphs [LNS11, BBL12, RM13, BPF14a, BHRD\*15, BKH\*16]. Figure 2.5 shows forest harvest data over 11 years. In information visualization, time juxtaposing is usually referred to as *small multiples* [CKN\*03], although small multiples are not necessarily built from time slices (see Figure 3(a)). Time juxtaposing has been also widely used for video summarization [TV07].

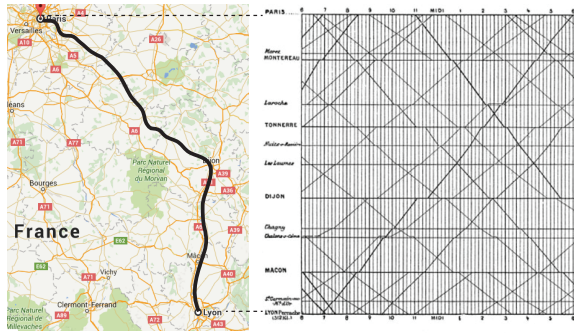


## 2.6. Space Cutting



**Figure 16:** The space cutting operation.

*Space cutting* consists in extracting a planar cut in a direction perpendicular to the data plane (Figure 16). An analogy is *slit-scan photography*, a process where a plate into which a slit has been cut is inserted in front of a camera and then moved while the film is being exposed [TGF08]. Slit-scan photography has been used to create special effects in movies, artwork and photo finishes in sports.



**Figure 17:** Example of space cutting (right): horizontal lines indicate train stops, vertical lines indicate times, and diagonal lines indicate moving trains [Mar78].

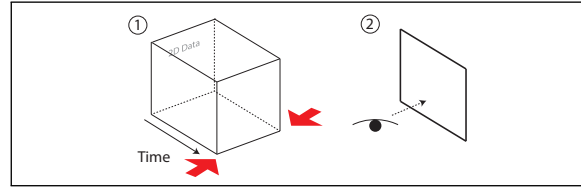
Space cutting has also been employed for visualizing temporal data. In the 19th century, Marey created a visualization of train connections between major French cities (Figure 17). This visualization can be described as space cutting on a geographical space-time cube, along the tracks connecting cities while diagonal lines indicate positions of trains at any time [Tuf86, Mar78].



**Figure 18:** Space cutting used to show road traffic [TGF08].

More recently, space cutting was shown to be useful for analyzing video logs [TGF08]: Figure 18 shows a space cut (called “tear” in the original work) extracted from a video space-time cube, and revealing traffic activity (car count, speed and direction) on a road. The time slice at  $t_1$  is shown to the left, together with the position of the segment extracted. The system is also able to show multiple longitudinal slices on top of each other (i.e. *space juxtaposing*).

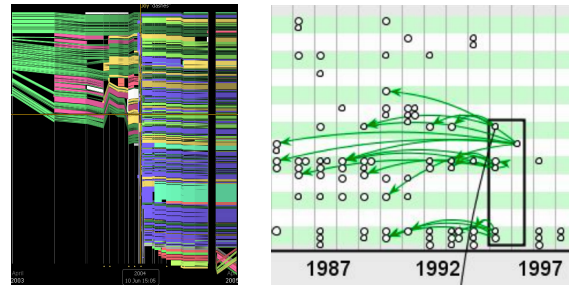
## 2.7. Space Flattening



**Figure 19:** The space flattening operation.

*Space flattening* is similar to space cutting, but involves flattening the cube along a particular direction on the data plane instead of extracting a cut (see Figure 19).

An example of use of space flattening in infovis is the History Flow technique for visualizing document histories [VWD04], illustrated in Figure 20(a). A space-time cube would consist of the different document version piled over time. The visualization shows the history of the article, built by collapsing each article revision into a one-pixel column, and then displaying all columns side-by-side. These operations are equivalent to flattening the article’s space-time cube along the x data axis. Colors correspond to contributors.



(a)

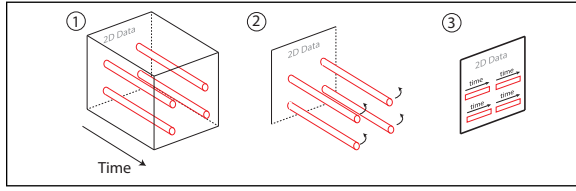
**Figure 20:** Examples for space-flattening. a) Edit history of a Wikipedia article [VWD04]. b) article citations over time [SA06, AS].

Space flattening has also been used for visualizing dynamic networks, based on node-link representations on the base plane [FBS06, SA06, BVB\*11]. For example, Figure 20(b) shows a screenshot from Semantic Substrates [SA06] where the y-axis is a 1D graph layout, and the x-axis shows when connections are established.

## 2.8. Repeated Drilling

*Repeated Drilling* is a more complex operation that consists in extracting drilling cores from a space time cube at several locations on the visualization plane, then rotating those cores in-place so they face the viewer (Figure 21).

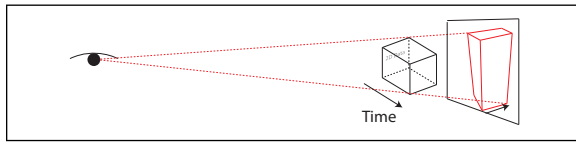
Two examples of drilling are mentioned in this article’s introduction (Figure 2). The top one shows the evolution



**Figure 21:** The repeated drilling operation.

of crime statistics in every US state [AA04] obtained from a geographical space-time cube containing a series of values for each state. The bottom one shows the evolution of high school population in several districts across three years [SRdJ05]. Although additional operations are involved (e.g., using silhouette graphs to encode values), both examples are conceptually based on repeated drilling. Repeated drilling has also been used in dynamic network visualization, for conveying changes in edge weight [BN11] and in attribute values [HSCW13].

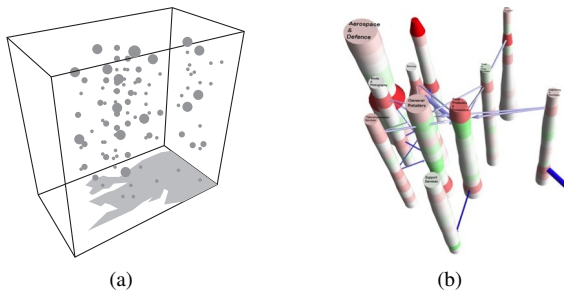
## 2.9. 3D Rendering



**Figure 22:** The 3D rendering operation.

3D rendering consists in showing a space-time cube the way three-dimensional objects are typically displayed on 2D media, i.e. by projecting it onto a 2D plane (Figure 22).

3D rendering is essentially a flattening operation but in contrast with time flattening and space flattening, it is (i) typically done on a plane not orthogonal to the cube's principal axes; (ii) can involve a non-orthographic projection (e.g., perspective projection); (iii) can involve 3D shading, i.e. the addition of light reflections and shadows.



**Figure 23:** Two examples of 3D rendering. (a) Occurrence of earthquakes (authors' illustration after [GAA04]), and (b) a dynamic Network [DG04]

In geography, 3D rendering has been used to visualize events such as earthquakes (Figure 23(a)), attributes

over time [TSWS05, TR09] or the movement of objects [Kra03, GAA04]. 3D rendering has also been used to visualize dynamic networks based on node-link diagrams; nodes become columns and links become bridges between columns [DG04, BC03] (Figure 23(b)). When the layout of the dynamic network also changes, nodes become worms [DE02, GHW09]. Worms also result from 2D scatterplots extruded into time [MR97].

## 3. The Design Space of Space-Time Cube Operations

The previous section reviewed several common operations that turn a conceptual time-space cube into a final two-dimensional visualization. Examples were selected for illustration, and the list is not meant to be exhaustive. Some of the operations were rather simple (e.g., *time cutting*), while others were more complex (e.g., *repeated drilling*) and could be described as a composition of several lower-level operations. Therefore, we provide in this section a more systematic description of the design space of space-time cube operations.

### 3.1. Basic Terminology





















A *space-time cube operation* takes a space-time object and produces another space-time object. A *space-time object* is a geometrical object within a space-time coordinate system (i.e. two dimensions for the visualization space and one for time). Possible space-time objects include: (i) *space-time volumes* (of which a complete space-time cube is an example), (ii) *space-time surfaces* (planar and non-planar), (iii) *space-time curves*, (iv) *points*, as well as (v) sets of disconnected volumes, surfaces, curves and points.








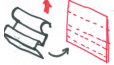














The ultimate goal of space-time cube operations is to transform a space-time cube into a space-time object whose shape is compatible with the shape of the media employed to convey the information. By media we mean the visualization's *physical presentation*, which is the physical object or apparatus that makes a visualization observable to the viewer [JD13]. In the vast majority of cases (i.e. computer displays and paper) the media has a planar shape.

For a given media, a space-time cube operation is *complete* if it takes space-time volumes as input and produces space-time objects whose shape match the media's shape. Otherwise, the operation is *incomplete*: it cannot be used to produce a valid visualization from a space-time cube. Several elementary space-time cube operations can be chained, in which case they form *compound operations*. A compound operation is complete if the first operation takes space-time volumes as input, and the last operation produces space-time objects whose shape is compatible with the media.

### 3.2. A Taxonomy of Elementary Space-time cube operations

A taxonomy of elementary space-time cube operations is shown in Figure 24 on the next page. The taxonomy breaks down space-time cube operations into four main classes:

Operations				Time	Space
Extraction	Point	Point Extraction			
	Curve	Planar Drilling	Orthogonal Drilling		
			Time Drilling	Space Drilling	
			Oblique Drilling		
		Planar Curvilinear Drilling			
	Non-Planar Drilling				
	Surface	Planar Cutting	Orthogonal Cutting		
			Time Cutting	Linear Space Cutting	
		Oblique Cutting			
		Non-Planar Cutting	Curvilinear Space Cutting		
			Other		
	Volume	Planar Chopping	Orthogonal Chopping		
			Time Chopping	Linear Space Chopping	
		Oblique Chopping			
		Non-Planar Chopping	Curvilinear Space Chopping		
			Other		
Flattening	Planar Flattening	Orthogonal Flattening			
		Time Flattening	Space Flattening		
	Oblique Flattening				
Non-Planar Flattening					

Operations			Time	Space
Geometry Transformation	Rigid Transformation	Translation		
		Time Shifting	Space Shifting	
	Rotation	Yaw		
		Pitch		
		Roll		
	Scaling			
Bending				
Unfolding				
Content Transformation	Filling	Orthogonal Interpolation		
		Time Interpolation	Space Interpolation	
	Volume Interpolation			
	Encoding	Recoloring		
			Time Coloring	Space Coloring
		Difference Coloring		
	Others			
	Labeling			
	Repositioning	Stabilizing		
		Bundling		
	Shading			
	Filtering			
	Aggregation			

**Figure 24:** Taxonomy of elementary space-time cube operations with schematic illustrations. Gray shading indicates non-leaves. The Time column regroups operations that are applied according to the time axis, while the Space column regroups operations that are applied according to the base plane.



- **Extraction** consists in selecting a subset of a space-time object (e.g., extracting a line or cut from a volume),
- **Flattening** consists in aggregating a space-time object into a lower-dimensional space-time object (e.g., projecting a volume onto a surface),
- **Geometry transformation** consists in transforming a space-time object spatially without change of content,
- **Content transformation** consists in changing the content of a space-time object without affecting its geometry.

The table in Figure 24 shows how general operations break down into more specific operations. On each of the two columns, general operations are on the left while more specific operations are on the right. Operations that are the most specialized (i.e., leaves on the taxonomy tree) are shown on a white background. Operations written in bold are those which produce planar surfaces, i.e. can be used as final operations on screen-based and paper-based media.

We quickly review the most specialized operations (white background), going from top to bottom on the left column, then on the right column. We also describe the parameters necessary to specify each space-time cube operation. Most of the operations have already been used in infovis, others have been added to fill gaps in the taxonomy.

- **Extraction:**

- **Point extraction** consists in selecting a specific point inside a space-time volume. This operation is defined by a 2D position on the data plane and a time value.
- **Time drilling** consists in extracting a line parallel with the time axis. It is uniquely specified by a 2D position on the data plane. For example, *repeated drilling* (Section 2.8) uses several drilling operations.
- **Space drilling** extracts a line perpendicular with the time axis. It is specified by a 2D line and a time value.
- **Oblique drilling** consists in extracting an arbitrarily oriented straight line from within a space-time volume.
- **Planar curvilinear drilling** consists in extracting a planar 3D curve from a space-time volume. This operation, as well as all operations above, is complete for 2D media.
- **Non-planar curvilinear drilling** consists in extracting an arbitrary 3D curve from a space-time volume. It is incomplete, and hence needs to be combined with other operations like *flattening* or *unfolding*. This operation can be used to extract object trajectories [KW04, RFF\*08].
- **Time cutting** consists in extracting a planar cut from a space-time volume in a direction orthogonal to the time axis (see Section 2.1). It takes as parameter a time value that defines the cut position on the time axis. It is a complete operation for 2D media.
- **Linear space cutting** consists in extracting a planar cut from a space-time volume in a direction orthogonal to the data plane (see Section 2.6). It is also complete, and takes as parameter a line or a segment parallel to

the data plane that once extruded over time defines the cutting surface.

- **Oblique cutting** consists in extracting a planar cut from a space-time volume that is neither orthogonal to the time axis, nor orthogonal to the data plane (e.g. [FLM00]). It takes as parameter a 3D cutting plane.
- **Curvilinear space cutting** is similar to linear space cutting except the cutting surface is produced by extruding a curve parallel to the data plane that is neither a line nor a segment. This operation produces non-planar space-time surfaces that further need to be flattened (e.g., using 3D rendering [TS12]) or unfolded (as in Figure 17).
- **Time chopping** is similar to time cutting but slices have a thickness instead of being infinitely thin. Since it produces volumes it is not complete for 2D media, and thus needs to be complemented with additional operations. It takes as parameter a time segment that defines the two cutting slabs (a slab is the infinite region between two planes).
- **Linear space chopping, oblique chopping and curvilinear space chopping** are similar to the previous cutting operations, with the difference that they produce volumes with a certain thickness instead of infinitely thin surfaces.

- **Flattening:**

- **Time flattening** aggregates a space-time volume into a plane orthogonal to the time axis (see Section 2.2). This operation takes as parameters a time value, a projection function and an aggregation function. The *projection function* maps 3D points to points on the plane. Examples include orthographic projection and perspective projection. The *aggregation function* describes how point values are combined. If values are defined in an RGBA color space, the function maps vectors of RGBA colors to a single RGBA color. Examples of such functions include alpha-blending (e.g., averaging all colors) and overplotting (i.e. only keeping the last color).
- **Space flattening, oblique flattening and non-planar flattening** are similar operations, but the surface on which the volume is projected is different (see previous cutting operations as well as Sections 2.7 and 2.9 for more details).

- **Geometry Transformation:**

- **Space shifting, time shifting, yaw, roll and pitch** consist in moving or rotating space-time objects. They can be used, e.g., for placing multiple cuts side-by-side or for rotating an entire space time cube rendered in 3D (e.g. [KW04, CCT\*99, BPF14b]).
- **Time scaling and space scaling** rescale space-time objects along their principal axes. They take as param-

eters one and two scalar values respectively, that define the scaling factor.

- **Bending** deforms space-time objects. For example, a space-time volume can be bent such that the time axis follows an arc instead of a line [DC03]. This operation takes as parameter a deformation function that maps 3D locations to 3D locations.
- **Unfolding** transforms a non-planar space-time surface into a planar space-time surface. An analogy is a map projection function that transforms a sphere or portion of sphere into a plane. An example of space-time unfolding is Maray’s train schedule (Figure 17), which can be seen as an unfolded curvilinear space cut performed on a time-evolving 2D map.

#### • Content Transformation:

- **Time coloring** consists in altering the colors of each time slice according to time. Examples include coloring each time slice uniformly according to a linear color scale (Figure 13), changing the hue of each time slice, or dividing the time axis in different regions and applying a discrete color scale (Figure 8).
- **Space coloring** alters the color of points in a space-time volume depending on their 2D position on the data plane.
- **Difference coloring** consists in altering the colors of each time slice according to the difference between time slices. One example is highlighting appearing nodes and disappearing nodes in a dynamic graph [RM13, BPF14a].
- **Time labeling** consists in adding time labels to each time slice or to objects inside a space-time volume (Figure 9(b)).
- **Stabilizing** consists in repositioning objects on each data plane so that their trajectories are as parallel as possible to the time axis. Examples include computing stable layouts for dynamic networks [AP12a], dynamic maps [HKV12], and stabilizing videos [BGPS07].
- **Bundling** consists in repositioning objects on each data plane in order to bring their trajectories closer to each other. One example is bundling air plane routes [HEF\*13].
- **Shading** consists in altering the color of a space-time volume’s content by simulating light propagation mechanisms (e.g., diffusion, specular reflection, drop shadows).
- **Filtering** consists in removing parts of a space-time volume’s content. One example is removing all points of a certain color or value [CCT\*99, DC03, BPF14b].
- **Aggregation** replaces multiple space-time objects by a single, larger space-time object. Different methods exist. For example, 3D kernel density estimation transforms a set of space-time points or space-time curves into 3D volumes or 2D (iso) surfaces [DV10].

### 3.3. Adaptive and Semantic Operations

So far we mostly described operations that are agnostic to the data and the content of the cube. *Adaptive operations* take into account the shape or content of the particular space-time objects they operate on. For example, an *adaptive time cutting* operation can be used to cut cubes according to regions with large changes instead of cutting them into regularly-spaced slices. This technique is used, for example, in adaptive video fast-forward [PJH05] and to highlight states and repetition in dynamic networks, using a technique called “Small MultiPiles” [BHRD\*15] (*time chopping + space shifting*). Similarly, an unfolding operation that works on any surface (as opposed to, e.g., only spheres), would be an *adaptive unfolding* operation.

*Semantic operations* take into account the data semantics of the space-time objects they operate on. One example would be a *semantic volume interpolation* operation that connects discrete sets of moving objects with lines or tubes (see Figures 9(b) and 23(b), as well as [Ros06, BPF14a]). This type of operation is semantic because it needs to know the identity of the objects to be able to match them on successive time slices. *Time labeling* operations such as the one used in Figure 9(b) are also semantic, because they need to know the location of datapoints of interest to place the labels appropriately. *Filtering, aggregation and bundling can also be seen as semantic operations* as well as all *difference coloring* and *space coloring* [VWD04, RFF\*08, BPF14b]. Finally, semantic operations can also be used to cut or chop cubes according in regular distances (e.g., hours, days, weeks). Flattening such time chops changes the temporal granularity of the cube.

### 3.4. Compound Operations

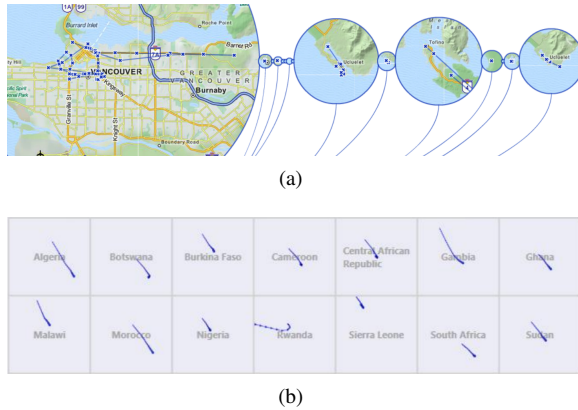
We previously defined *compound operations* as several operations applied in sequence. According to our taxonomy from Figure 24, some of the operations we introduced in Section 2 are elementary, namely *time cutting*, *time flattening*, *space flattening*. Others are compound and can be broken down as indicated in Table 1. In our notation, the symbol  $+$  refers to a composition, the symbol  $*$  refers to operations applied multiple times and the symbols  $[ ]$  refer to optional operations.

A compound operation inherits the parameters of its sub-operations. For example, a *discrete time flattening* operation is specified by a sequence of time values, as well as a projection function and an aggregation function. But in practice, most compound operations enforce constraints between their parameters. For example, all *space scalings* from a *time juxtaposing* operation are typically the same. Possibly those parameters could even result from a function. For example distance between individual time cuts in a *discrete time-flattening* could follow a logarithmic distribution, to give more importance to recent time steps.

Many elaborate temporal data visualization techniques

can be described as compound operations. For example, the “Visits” technique (Figure 25(a)) shows the history of a person’s travels using *semantic (time chopping + space chopping + time flattening + space shifting)\**. The conceptual space-time cube captures the location of the person in the entire world over a period of time. Each circle in Figure 25(a) is a *time-flattened* segment from this space-time cube. Each segment has also been *space-chopped* to show only a portion of the entire world. Time-flattened segments are then *shifted* on the data plane to distinguish them visually and to reflect the temporal order in which the regions have been visited.

A further example of a compound operation is shown in Figure 25(b). The technique is similar to that of Figure 3(a) and shows one connected scatterplot per country, visualizing infant mortality vs. life expectancy over time. While the original article [RFF\*08] calls this technique “Small multiples”, our framework would describe it as the following compound operation: *(filtering + interpolation + time flattening + space shifting)\**. That means, the initial space-time cube consists of a 2D scatterplot containing all countries, and time as third dimension. For each country, the technique extracts all the corresponding points, interpolates between them, flattens them across time, and finally juxtaposes the resulting space-time surfaces.



**Figure 25:** Examples for compound operations: (a) in *Visits* [TBC13] (b) in a study by Robertson et al. [RFF\*08]

Compound Operation	Elementary Operations
Discrete time flattening	time cutting* + time flattening
Colored time flattening	time coloring + time flattening
Time juxtaposing	(time cutting + space scaling + space shifting)* + time flattening
Marey’s schedule	curvilinear space cutting + yaw + unfolding
Slit tears	(linear/curvilinear space cutting + yaw + [unfolding] + space shifting)*
Repeated drilling	(time drilling + time scaling + yaw)*
3D rendering	[shading] + oblique flattening

**Table 1:** Example compound operations decomposed.

### 3.5. Dynamic Operations

So far we only considered operations (both elementary and compound) that transform a space-time cube into a static visual representation. On computer displays, operations can also be applied in a dynamic manner. Dynamic operations can involve either animation or interaction.

#### 3.5.1. Animation

We refer to *animation* as the process of applying different operations on a space-time cube over time, or similarly, varying the parameters of an operation over time.

The most common form of animation consists in changing the position of a cutting operation over time, i.e. *animated time cutting*. This results in the space-time cube content being “played back”. For example, if the space-time cube represents a visual scene like video surveillance data, synchronizing the motion of the slice with a clock will result in a real-time playback of the original scene. When significant data is skipped during playback, the animation is closer to a *discrete time juxtaposing* operation, except slices are shown in sequence instead of being laid out side-by-side.

An animated time cutting operation can be preceded by a *filling* operation in order to produce smooth animated transitions. Many examples exist in the literature, for example when animating dynamic networks [ATMS\*11, RM13, BPF14a] or scatterplots [Ros06, RFF\*08]. Most of these examples can be described as *semantic volume interpolation + animated time cutting* operations. Animated time cutting can also be combined with other space-time cube operations such as *time flattening*. For example, Gapminder can combine scatterplot animations with static trails for points of interest (a *filtering + time flattening* operation) [Ros06].

While many animation techniques can be described as animated time cutting on *static* space-time cubes, more elaborate techniques require operations to be applied in real-time. For example, Hurter et al.’s system [HEF\*13] uses *animated time chopping* to animate a network over time while preserving temporal context information. At every animation frame, a *time flattening* is applied that produces colored trails and a dynamic *bundling* operation is applied that guarantees a continuous animation without jumping bundles [HET12].

Although *animated time cutting* and its many variants are the most common forms of animation, other animated operations exist. For example, *animated 3D rendering* can explain a transition between two space-time cube operations to a user by smoothly rotating a space-time cube representation [BPF14b]. This technique will be discussed in Section 6, where we review space-time cube visualization systems.

#### 3.5.2. Interaction

*Interaction* is similar to animation, except the changes in the space-time cube operations are under the user’s control.

Consider *animated time cutting*: if the position of the cutting plane is controlled by the user (e.g., by dragging a slider) instead of being automatically moved, then the operation becomes *interactive time cutting*. A common implementation of interactive time cutting is the seeker bar on a video player. As with animations, any operation can be made interactive. Examples of interactive operations abound, and we will review some of them in the next section.

Another example of interaction is *interactive curvilinear space-cutting + oblique flattening*. Tominski and Schulz [TS12] allow user to draw a curve onto a map, then show the curvilinear space cut in a perspective projection.

#### 4. The Inner Structure of Space-Time Cubes

So far, we mostly considered the generalized space-time cube as monolithic entity and operations were defined on the entire cube. However, two space-time cubes can look quite different (see Figures 23, 29, 30, 32, 33), depending on the characteristics of the data being visualized; migration of animals, earthquakes, changes in vegetation, dynamic networks, surveillance videos, or scatterplots evolving over time. Such differences define how the space-time cube “looks”, and which we refer to as its *inner structure*.

On the other hand, different data sets can result in similar inner structures. For example networks in the form of node-link diagrams, and multivariate data visualized as scatterplots over time, both yield structures similar to the first row in Figure 26. The inner structure of a space-time cube is an important factor when choosing between space-time cube operations and is independent of the specific type of data. This allows for a general discussion about the advantages and disadvantages of an operation, as well as to transfer visualization techniques between domains.

##### 4.1. Decomposing Space-Time Cubes

We assume that any space-time cube can be subdivided into lower-level space-time objects (see Section 3.1). A common example is a set of 3D curves or tubes generated by moving objects such as cars, persons, nodes in a network, or points in a scatterplot (see Figure 26 ①). These curves can yield different patterns depending on the data ②③④. Another typical structure is obtained from dense matrix data such as videos or dynamic heatmap visualizations. For these datasets, the object of interest (e.g., a pixel) is fixed in space but its color changes over time. The space-time objects are therefore multicolored lines parallel with the time axis and with no intervening space ⑤. Again, different patterns of variation are possible ⑥.

A space-time cube can be subdivided in different ways. For example, pixels with an alpha channel can yield higher-level space-time objects with intervening spaces: ⑦ could be obtained by filtering a pixel-based landscape visualization according to the type of vegetation while ⑧ could be

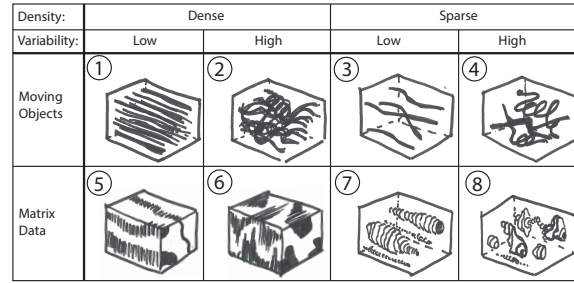


Figure 26: Different inner structures for a space-time cube.

obtained by isolating bush fires [CCT\*99]. Alternatively, optical flow can be extracted from a video, which would yield a structure more similar to ①. Conversely, any animated visualization could be turned into a video and yield a structure similar to ⑤. These are all different ways of looking at a space-time cube. Ideally, however, the focus should not be on the specific decomposition used, but rather on the overall properties of the inner structure.

##### 4.2. Dimensions of Inner Structures

Figure 26 only shows eight possible examples of inner structures, but many more structures are possible. For example, moving objects can also change color or shape over time, or disappear and reappear, and this would yield different structures than the ones shown on the Figure. Four dimensions are however enough to capture the most important properties of inner structures, namely: *density*, *variability in positions*, *variability in visual attributes*, and *object lifespan*.

**Density:** The density of a space-time cube refers to how much non-empty space it contains. The inner structure of a space-time cube can be *dense* (Figure 26 ①②⑤⑥) or *sparse* (Figure 26 ③④⑦⑧). Datasets like videos (Figure 26 ⑤⑥) produce inner structures that are maximally dense, and which will be referred to as *complete*. Furthermore, a space-time cube can be *spatially dense* but *temporally sparse* (e.g., a few photo snapshots), or vice versa (e.g., a GPS log over several days). Density is a key property of inner structures, and it is the only dimension that is independent from the decomposition used.

**Variability in positions:** This dimension describes the amount of motion that 2D objects undergo in the space-time cube, which in turn affects the shape of the 3D space-time objects that make up its inner structure. For example, case ③ in Figure 26 (which could be moving people) involves rather straight curves, while case ④ (which could be animal movements) involves spaghetti-like curves. Matrix data (cases ⑤ to ⑧) yield zero variability in positions, since pixels are fixed and space-time objects are lines parallel with the time axis. Using an optical flow decomposition, case ⑥ would exhibit more variability in positions than case ⑤.



**Variability in visual attributes:** This dimension refers to the amount of changes (color and shape) that 2D objects undergo in the space-time cube. For example the pixels in ⑤ and ⑥ undergo changes in color, with a higher variability in ⑥. In contrast, the moving objects in ① to ④ have a fixed shape and color, and therefore the cube's inner structure exhibits no variability in visual attributes. In a multivariate network [KPW13] setting, this variation could exist, e.g., the objects were moving taxis and their color or size (e.g., radius) encoded the number of customers they carry. Case ⑧, which involves blobs moving over time (e.g., a spreading bush fire), yields high variability while case ⑦ does not.

**Object lifespan:** Object lifespan refers to how long 2D space-time objects persist on average, which has an influence on the portion of the time axis covered by the 3D space-time objects. The lifespan is maximum when all objects span the entire time axis (Figure 26 ① to ④). Regular matrix data also yields a maximum lifespan (Figure 26 ⑤ and ⑥). In contrast, objects can reflect events that have a start and an end: ⑧ involves short object lifespans whereas ⑦ involves relatively long object lifespans. Events without duration produce the shortest possible lifespans, and yield inner structures consisting of points. Since objects do not persist, there is no variability in positions or visual attributes.

The inner structure of any space-time cube can be described along the four previous dimensions. For example, the earthquake cube in Figure 23(a) is relatively sparse and has minimal object lifespan. The dynamic network cube in Figure 23(b) is also relatively sparse, but has various lifespans (long for nodes and minimal for links). Also, nodes are columns that exhibit no variability in position but a high variability in visual attributes (color).

Thereby, it does not matter whether the initial data is discrete or continuous in time or space. Discrete and continuous are two complementary aspects, both encompassed by the generalized space-time cube. For example, *repeated (time chopping + time flattening)* can transform a continuous time dimension into a discrete time dimension. On the other hand, *time interpolation* can yield a continuous impression of otherwise discrete data.

In the next section we will see why a dataset's characteristics matter, and how the inner structure of a space-time cube can be altered with space-time operations.

## 5. Which Operations Work Best?

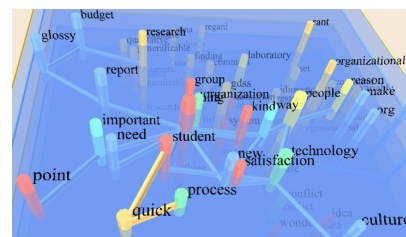
So far we essentially focused on the descriptive function of our framework: the different types of operations, how they can be combined or made dynamic, and the inner structure of space-time cubes. We now have all the elements to discuss practical strengths and weaknesses of space-time cube operations. This aims to support the choice of operations when designing visualizations, to researching novel visualization

techniques to leverage the strengths and overcome the drawbacks of space-time cube operations.

Strengths and weaknesses of operations are directly related to user tasks, since certain operations yield visualizations that may better suit a certain task. So far, only few user studies compare space-time cube operations but will be used to ground our discussion. However, only a small subset of all possible operations have been covered by user studies; many of the existing findings are hard to generalize beyond the domain, type of data, and operation parameters used in the stimuli. Thanks to the ability of our framework to reduce complex techniques into elementary operations, useful advice can often be provided based on more general knowledge on visual perception and on common sense. With respect to data type, the inner structure of a space-time cube can in many cases inform the suitability of certain operations. For example, if a dataset produces maximally dense space-time cubes, a 3D representation will always be problematic unless other operations (filtering, cutting) are used.

### 5.1. 3D Rendering

*3D rendering ([shading] + oblique flattening)* is the operation most commonly associated with the space-time cube concept, so it seems necessary to start with a cautionary note. 3D visualizations are known to pose major difficulties to users and are overall not recommended [Shn03, Mun15]. Difficulties include (i) occlusion, (ii) depth ambiguity, (iii) perspective distortion, (iv) color distortion and (v) navigation difficulties, further discussed below. *3D rendering* can however be useful for providing a general overview of a space-time cube's inner structure (density, variability and object lifespan). It is also the most effective approach when the goal is to have observers explicitly think in terms of a space-time cube, as it is the case for most diagrams in this article.



**Figure 27:** A time-evolving network rendered in 3D with translucency [BC03].

Occlusion issues are due to the use of a *flattening* operation together with an *overplotting* aggregation function (i.e. near objects are drawn on top of far objects). The amount of occlusion depends on the *density* of the inner structure (see Figure 27). Therefore, *3D rendering* is mostly adapted to *sparse* structures, as in Figure 23. Additional space-time cube operations can however be employed for visualizing *dense* (and sometimes *complete*) structures. An effective

technique involves sparsifying the cube's structure using *interactive filtering* [CFC\*96, BPF14b]. A related approach involves the use of *translucency filtering* or more generally, translucent space-time objects (Figure 26). However, translucency produces visual clutter without fully addressing the problem of occlusion, and we found few instances of this technique yielding legible results. Also, occlusion is an important depth cue [WFG92] that is mostly lost when translucency is employed. For similar reasons, we recommend against making the whole space-time cube translucent – or equivalently, using an *alpha blending* aggregation function. Elaborate interactive operations exist that can decompose a space-time cube using *extraction* and *rigid transformation* operations, and allow users to look inside [CCF96].

In cases where space-time objects are densely clustered around certain areas with the space-time cube, *aggregation* can help reducing visual complexity and occlusion. In order to avoid rigid objects inside the cube, clusters of space-time cube objects can be estimated through transparency [DV10].

The *flattening* operation is further responsible for causing depth ambiguity. On Figure 23(a) for example, it is impossible to extract the spatio-temporal coordinates of the points. In the real world, most depth ambiguities are resolved through stereoscopic vision and structure from motion [Tod04, WFG92]. Structure from motion is as powerful as stereoscopic vision [Tod04], and this is why *3D rendering* is the most effective when the space-time cube is allowed to rotate through *animated* or *interactive 3D rendering*. Incidentally, being able to “look behind” space-time objects also mitigates occlusion problems. When interaction cannot be used (e.g., on paper media), static depth cues are crucial. One approach for structures with no or very low *variability in positions* is to have space-time objects “touch” the ground. This is already the case for structures with maximum *object lifespans* such as in Figure 23(b). For short *object lifespans* like in Figure 23(a), it is advised to either add lines connecting objects to the ground [KW04] or add drop shadows [WFG92, TKB07] by applying a *shading* operation with perpendicular lighting.

Perspective distortion is due to the use of a *perspective* projection function: far objects appear smaller than near objects, making it difficult to accurately compare sizes and lengths. This issue can be eliminated with the use of an *orthographic* projection function instead. However, doing so will eliminate an important depth cue and may produce depth reversals analogous to the necker cube illusion [WFG92]. Again, this type of ambiguity can be alleviated by the use of other depth cues such as occlusion, contact and *shading*.

Color distortion is a side-effect of the *shading* operation, since it alters the color of the space-time objects. This is not a problem when color is not encoding information or when simple color scales are used as in Figure 23(b), but could be an issue when numerical data is encoded through brightness.

Navigation difficulties with 3D visualizations are due to

the *oblique flattening* operation having too many degrees of freedom. One way to facilitate navigation is to add constraints, i.e. only let users manipulate *3D renderings* in ways that are essential to their task [SSS\*01, Shn03].

Generally speaking, space-time cubes are a convenient conceptual tool for thinking about temporal visualizations, but explicitly showing them using *3D rendering* poses a number of problems and should be done only when necessary (e.g. for showing an overview or for explanatory purposes). Computer screens and paper are 2D media, and are optimized for 2D visualizations [Shn03]. As we have already seen, many 2D solutions exist to show temporal data.

## 5.2. Time Flattening

*Time flattening* (Figure 7) can be seen as a 2D technique, or as a specific form of *3D rendering* where ambiguities are purposeful and animation or interaction are not necessary to make sense of the data. *Time flattening* essentially turns a space-time cube into a spatial visualization: each point on the resulting image shows the history of this point aggregated over time. As a result, spatial relationships and trends are preserved and are clearly visible, such as the path of Napoleon's army in Figure 8 or the distribution of data values in Figure 9(b). Any aggregation function can be used, e.g., alpha-blending for showing averages, or more elaborate techniques for showing cumulative statistics (Figure 9(a)).

The major drawback of *time flattening* is that it reduces the time axis to a point (Figure 7), so most temporal information is lost, including the ordering of events, temporal intervals, and absolute timestamps. However, part of this information can sometimes be cognitively reconstructed from object trajectories. For example, when reading Napoleon's march (Figure 8), one knows that an army does not jump between random locations but progresses smoothly, so not all orderings of events are plausible. Knowing that an army generally shrinks in size is then enough to infer its direction of motion: eastward then westward. The assumption of motion smoothness also helps to read the infographics of Figure 11: the ice skater either progresses from left to right, or from right to left. Basic knowledge about how a human body moves then suffices to entirely reconstruct the direction of motion and the ordering of events.

For flattened trajectories to be legible and informative, the space-time cube needs to be *spatially sparse*, with long *object lifespans*, and exhibit a reasonable amount of *variability in positions* and in *visual attributes*. A space-time cube can sometimes be processed to meet these criteria. For example, *time coloring* (Figure 13) is a way of adding *variability in visual attributes*. This operation has the additional advantage of explicitly encoding time. However, color is a poor visual attribute since the human visual system cannot discriminate between many colors [CAS05], and no color scale exists that naturally maps to time. For example, red in Figure 13(a) represents old events, while it represents new events in Figure

13(b). If time intervals (or object speeds) are more important than absolute times, a more effective approach is to change the color or shape of objects at regular time intervals.

Additional operations can be applied to a space-time cube in order to reintroduce missing time information. For example, the Phillips curve in Figure 9(b) uses *semantic volume interpolation* to connect points and show their ordering. Both Figure 9(b) and Figure 8 (for the army's return trip) employ *time labelling* to explicitly convey event timestamps.

*Discrete time flattening* (*time cutting*\* + *time flattening*) is a variant that can also clearly expose time intervals and object speeds. Since this technique can result in loss of information, it works best for structures that are *sparse in time*. The way time cuts are selected is important. Equidistant cuts generally expose relative times, durations, and rates of change. When not all cuts are equally informative, non-equidistant cuts may be preferred, but at the cost of losing most of this temporal information. If cuts are chosen so that objects do not overlap, these objects become clearly visible. This technique is used in Figure 11 to expose the skater's posture, an information that helps to read the visualization.

In summary, *time flattening* is an operation that produces easy-to-interpret visualizations: space simply maps to space. It is useful for showing data patterns aggregated over time and where space is more important than time. But since it collapses the time axis into a point, it is most powerful when it can leverage humans' ability to reconstruct time from trajectories. This often requires the use of additional space-time cube operations and a very careful visual design.

### 5.3. Space Flattening

*Space flattening* (Figure 19) is another form of *orthogonal flattening* that avoids some of the drawbacks of regular *3D rendering*. The orthogonal projection function creates a meaningful spatial encoding, since one axis maps to time while the other axis maps to space. The resulting visualization can be read as a regular timeline.

*Space flattening* is effective when both time and space are important, but one dimension of space is more informative than the other. This is the case for edit patterns in Wikipedia articles (Figure 20(a)). This technique is also useful for focusing on one data dimension at a time, when the two data axes have clearly different roles (e.g., in a 2D scatterplot). However, it produces visualizations that are generally less easy to interpret if the base plane does not have a natural coordinates system (e.g., in a node-link diagram or on a map).

Other cases require a linear ordering of data elements along the single remaining spatial dimension, in order to avoid occlusion. Such a linear ordering can be obtained by *unfolding* as in Marey's train schedule (Figure 17). In networks, a linear ordering of nodes can help reduce visual clutter caused by edge crossings. An ordering can be defined

based on different criteria and strongly influence the overall readability of the visualization, as well as emphasize particular patterns [vdEHBvW13].

### 5.4. Time Cutting

So far we covered different types of *flattening* techniques. These techniques essentially aggregate the data in a space-time cube and are helpful for overview tasks, but fail at providing details. We now turn to *extraction* techniques. Among these techniques, *time cutting* is by far the most common. A single time cut provides a clean and fully detailed snapshot of a particular time stamp. It can potentially fill the entire screen and show many details, and is furthermore free of using any additional visual variable for time (e.g. color, brightness). A simple *time labeling* of the time cut can be enough.

However, since the goal of a temporal visualization is to show information across time, more than one time cut must be shown. We already mentioned *discrete time flattening* as a possible approach, although it still suffers from the drawbacks of aggregation. Two aggregation-free alternatives are *time juxtaposing* and *animated time cutting*. We review them separately, then discuss empirical research comparing both.

### 5.5. Time Juxtaposing

*Time juxtaposing* (*(time cutting + space scaling + space shifting)\* + time flattening*) allows to show different time slices simultaneously. Due to our exposure to sequential art, it produces familiar visualizations, that are easy to interpret. Usually, *space shifting* follows the occidental convention of laying out time slices either linearly from left to right, or on 2D from left to right then top to bottom [McC94].

The main advantage of *time juxtaposing* over *flattening* is the legibility of individual time slices due to the absence of occlusion. However, the *space scaling* operation also has the effect of shrinking time slices: the more slices are shown, the less space – and therefore resolution – every slice has. The *time juxtaposing* operation therefore requires finding the best trade-off between spatial and temporal resolution.

A potentially serious drawback of *time juxtaposing* is that time slices use inconsistent spatial coordinate systems, due to the *space shifting* operation. This can make it difficult to relate moving objects across time slices and reconstructing their trajectories. However, if objects on a time slice can be visually distinguished and have a consistent appearance across time (i.e. low *variability in visual attributes* and in *positions*), relating time slices becomes easier. With few objects and pre-attentive visual encodings [War12], relating slices can be effortless. If there are many objects in each time slice and/or high variability in positions and visual attributes, tasks may still require visual search and split attention. However, this effort can be alleviated through interaction by adding brushing and linking support [BC87].





**Figure 28:** Territory of later Yugoslavia (red line).

*Time juxtaposing* is analogous to *discrete time flattening*, except time slices are laid out next to each other instead of being blended into a single image. As a result, the technique yields similar benefits compared to a regular *time flattening*. One is that it is able to expose the temporal information that is otherwise lost by aggregation, including event ordering, time intervals, speeds and absolute time stamps. But again, this assumes that objects can be easily related across slices, which can be difficult when slices do not share the same co-ordinates system.

Like with *discrete time flattening*, the proper choice of time slices is crucial with *time juxtaposing*, with the difference that granularity also affects spatial resolution. Again, equidistant cuts are better at preserving temporal information. But when not all cuts are equally informative, non-equidistant cuts (i.e. *semantic time cutting*) may be preferred. For example, when showing changes in territory, historical atlases and newspapers often employ this technique to emphasize different periods and historical events (Figure 28). Individual dates typically need to be added (i.e. *time labelling*) to reintroduce the missing time information.

## 5.6. Animated Time Cutting

A popular way of exploiting *time cutting* is through *animated time cutting*. This approach inherits the benefits of *time cutting* compared to *flattening* in that there is no aggregation. Therefore, individual time slices can be seen clearly, without occlusion. In addition, *animated time cutting* is probably the most “natural” and easiest to interpret among all possible techniques, since space maps to space and time maps to time. Animations can also be visually appealing and users can find them engaging.

In contrast with *time juxtaposing*, *animated time cutting* does not suffer from the reduced spatial resolution due to *space scaling*: each time slice can span the entire screen. Also, many time slices can be shown in a short period of time (e.g., 60 in only one second on a 60Hz display). However, fast animations are not necessarily easy to comprehend. If a space-time cube’s structure is *temporally dense* (e.g., a long surveillance video), an animation can take time to be perceived and understood [TMB02]. As with previous techniques, *semantic time cutting* can be used to speed up the playback when not all time cuts are equally informative [PJH05]. Nevertheless, like time juxtaposing, the technique may not be suitable for conveying information with

extremely high temporal density and aggregation (e.g., in the form of time flattening) may be preferable.

One major drawback of animated time cutting is that time slices must be presented in a predefined (typically chronological) order. As a result, the user must rely on memory in order to fully understand the data [Mun15]. Since time slices are not shown simultaneously, they cannot be freely accessed [TMB02]. This makes it difficult to carry out many types of exploratory tasks, such as comparing two non-successive slices, or examining a slice in context with other slices. Annotations are also impossible. These are serious issues that can be partly alleviated by using *interactive time slicing* instead of *animated time slicing* and allow to animate between non-adjacent time cuts [BPF14a]. But these techniques do not preclude the use of memorization. Also, both *animated* and *interactive time slicing* usually requires dynamic displays (though one can think of physical flipbooks), whereas all previous methods are compatible with actual static media such as newspapers.

Nevertheless, *animated time cutting* can be very effective at revealing changes between successive slices, as our visual system is equipped for pre-attentively detecting in-place changes [Bar97]. It can reveal subtle visual changes that are hard or impossible to detect otherwise [WB05, GMH\*06], as well as large and drastic visual changes. Our perceptual ability to detect and interpret changes, however, requires the absence of visual disturbances (e.g., flicker) between time slices of interest [Ren02], and ideally, that successive time slices are similar enough to be visually integrated and perceived as a motion [FT06]. This means the *variability in positions* should not be too high as to produce large discontinuities. If it is the case, *time interpolation* or *volume interpolation* need to be used to produce smooth animated transitions. Changes can also be highlighted (e.g., through a *difference coloring* operation [APP10, RM13, BPF14a]) in order to facilitate their detection and interpretation.

## 5.7. Stabilization

The *stabilization* operation can be used to complement several of the previously seen approaches. While *interpolation* allows to produce smooth transitions between predefined time slices, *stabilization* allows to reduce *variability in positions* when positions do not encode data. Although it may reduce the legibility of trajectories when using *time flattening*, this technique may facilitate object tracking tasks in *animated time cutting* and object mapping tasks in *time juxtaposing*. It is mostly used in dynamic graph drawing. The goal of dynamic graph drawing is to compute optimal layouts for time-evolving networks by either partially stabilizing nodes, or by computing a fixed global layout (Figure 23(b)). The underlying motivation is to allow users to maintain a spatial “mental map” of the data [ELMS, MELS95, AP13].

## 5.8. Empirical Evidence

There has been a considerable amount of literature in information visualization, cartography and educational psychology about the respective merits and weaknesses of animated vs. static representations [TMB02]. The scientific debate essentially opposes *time juxtaposing* with *interactive time cutting*. Other studies, mostly from the domain of geo-spatial visualization, compared *time flattening* with *3D rendering* and sometimes with *animated time cutting*.

Overall, studies suggest that *time juxtaposing* can be effective for many tasks as long as objects of interest can be easily related across time slices or visually indicated through linked highlighting. It has been shown to be faster and more accurate than *interactive time cutting* for understanding temporal trends in dynamic scatterplots [RFF\*08]. For dynamic network analysis, it has been consistently shown to be faster for a number of tasks (e.g., reading change in node degree or number), with no measurable difference in error rates [APP11, FQ11]. Thus, for most tasks, time juxtaposing has been shown to be faster than interactive animation with no impact on the error rate [TMB02].

However, in specific circumstances, *interactive time cutting* has been shown to bring advantages. For example, it yields less errors when detecting the simultaneous appearance of nodes or edges in a network visualization, although it is slower [APP11]. *Animated time cutting* is very effective for highlighting subsets of data against a similarly looking background. Ware and Bobrow [WB05] demonstrate this quite convincingly for highlighting subgraphs in a large graph, and Griffen *et al.* find that it helps identify moving clusters against a background with identical colors [GMH\*06]. In both cases, animation was both more accurate and faster. There is also some evidence that participants use both spatial and motion information when memorizing animated network sequences, suggesting that animation can help with the memorization of dynamic data [AP12b]. Thus, *interactive time cutting* and *animated time cutting* seem useful for highlighting data subsets (using motion) and for representing a small number of dynamic changes. Short animations, consisting of just 2-3 time steps, may be useful for clarifying drastic reorientations in the visual representation of the data [TMB02]. There is some evidence to support this finding in the context of dynamic graphs where the stability of the drawing is low and important nodes in the task cannot be highlighted throughout the time series [AP16].

Concerning the efficiency of *3D rendering*, results are also mixed. On a data set of four people moving through space, Kristensson *et al.* [KDA\*09] found that *time flattening* with *space coloring* (each person had an individual color) performed better for simple tasks than *3D rendering* with *shading*. However, the latter was found to better support more complex overview tasks. Investigating similar techniques but on a much larger data set (203 trajectories), Kveladze *et*

*al.* [KK12] conducted a participatory design session with experts in geography. Experts reported that *shading* and mapping data attributes to trajectory thickness cluttered the visualization. Instead, they required *space coloring* in order to better distinguish people. *Time flattening* was generally perceived too overloaded. However, experts further reported that *3D rendering* “helps in earlier stages of the research to get some hints about the data”.

Willems *et al.* [WvdWvW11] compared *non-interactive animated time cutting* with *3D rendering* and an enhanced version of *time flattening* [WvdWvW09]. Results show that overall, each technique was best for one task. The study discusses many exceptions, depending on data *density*, and *variability in position*. *Animated time cutting* was reported to be the least robust to a *high variability in position*. For the number of objects, Willems *et al.* report that *3D rendering* was least scalable, while *animated time cutting* performed best. Comparing *interactive animated time cutting*, *time juxtaposing*, and *aggregation (iso-surfaces)* in a space-time cube, Brunsdon *et al.* [BCH07] came to similar results.

The *stabilization* operation has also been empirically studied. Many studies found no positive effect [PS08, SP08, APP11, AP12b]. However, studies on directed graphs [PHG06] and recent studies on general, undirected graphs [GEY12, AP12a] have demonstrated benefits on mental map preservation. Overall, stabilization can help for a number of tasks, but further study is needed.

Overall, previous empirical studies paint a complex picture, and several controversies such as static vs. animated and 2D vs. 3D are far from being settled. Many studies emphasize task as a main source of trade-off. But most likely, the respective advantages of different approaches also heavily depend on the inner structure of space-time cubes and on the specific space-time cube operations being used. For example, employing smooth transitions (an *interpolation* operation) and giving subjects control over animations (*interactive time cutting*) can have a huge impact on the effectiveness of animations. Also, different datasets may produce widely different *inner structures*, and dramatically influence the efficacy of techniques. By clarifying these differences and providing a terminology to describe them, we hope that our framework will help design more informative user studies and eventually settle ongoing debates.

## 5.9. Other operations

We discussed the benefits and drawbacks of space-time cube operations commonly used for visualizing temporal data. As suggested by our framework (the elementary operation taxonomy in Figure 24 and the different ways to combine them), a possibly infinite variety of operations exist and it is impossible to analyze them all. It is likely that among all possible operations – especially the least constrained ones such as *oblique* and *non-planar* operations – the vast majority will be harder to read and to interpret. Popular approaches

are particular in that they mostly involve orthogonal operations, especially along the time axis (i.e. *time flattening* and *time cutting*). One striking exception is *3D rendering*, which combines several complex operations to emulate how we perceive solid objects in the real world (e.g., rotation, perspective projection, shading) and produces familiar (but sometimes illegible) images.

It is however also likely that the immense design space of space-time cube operations contains techniques that need to be learned, but can yield unique benefits for specific temporal data analysis tasks that usefully complement standard approaches. Some of these have already been discovered (see, e.g., Figures 2, 5, 2.5, 17, 18, 20(a), 31). The exciting possibility that the design space contains a number other “pearls” to be discovered stresses the importance of research in novel temporal visualization techniques. By discussing the multiple trade-offs involved in known techniques and explaining how these trade-offs originate from the intrinsic properties of elementary operations, we hope that our framework will be able to guide designers and accelerate future discoveries. Another promising line of work consists in supporting multiple space-time cube operations, and is discussed next.

## 6. Space-Time Cube Systems

Choosing an appropriate space-time cube operation depends on many factors and almost always involves tradeoffs. In this section we review a representative sample of visualization systems that address this issue by supporting multiple space-time cube operations. Those systems almost invariably use *3D rendering* as an explicit representation of the space time cube, both for showing an overview and for explaining how different operations relate. Operations are implemented on the visual model of the cube. We call these systems *space-time cube systems* and the traditional understanding of space-time cube in geo-visualization is in fact a geographical space-time cube system. Because space-time cube systems work by letting people switch between different operations and tune their parameters, *interaction* and *animation* are central features.

### 6.1. GeoTime

GeoTime is a carefully-designed commercial system for analyzing spatio-temporal data [geo, KW04]. Events are shown as spheres on a *3D rendering* view that can be freely rotated (Figure 29(a)). This view also uses a reference plane, and a *semantic volume interpolation* operation is applied to indicate event ordering. Users can perform *time chopping* operations by dragging on a timeline widget. GeoTime also supports *time flattening* and *space flattening*. Figure 29(a) shows a *space flattening* view where time runs from top to bottom, and a reference plane is provided that can be rotated. Thin gray lines connect the two views. Finally, pan & zoom is supported through *space chopping* + *space scaling*.

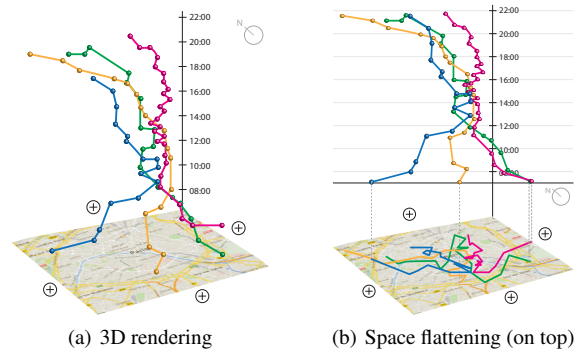


Figure 29: Illustration after GeoTime [geo]

### 6.2. Tardis

Tardis [CCT\*99, CFC\*96] is a system for visualizing environmental data using *3D rendering* in combination with advanced space-time cube operations. The voxels in the cube are color-coded depending on the type of vegetation, its age, soil characteristics or the presence of bush fires (Figure 30).

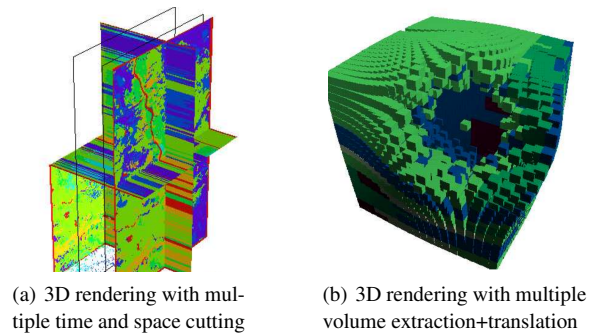
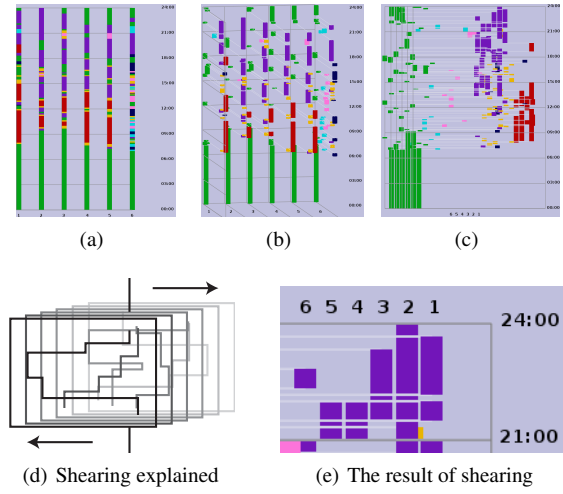


Figure 30: Tardis [CCT\*99] and visual access [CFC\*96]

Tardis implements *interactive semantic filtering*: users can, e.g., select a particular type of vegetation or a range of vegetation ages. In addition, Tardis supports *interactive orthogonal cutting*, but in contrast with our previous examples, cutting is always used in combination with *3D rendering*. Users can define and manipulate multiple orthogonal cutting planes (Figure 30(a)). Further operations include opening the cube like a book (*interactive (volume extraction + rotation)\**) or apply a 3D fisheye effect (*interactive (volume extraction + translation)\**), pushing away voxels from the cursor (“Visual Access Distortion”, Figure 30(b)).

### 6.3. VISUAL-TimePacTS

VISUAL-TimePacTS is a system for analyzing activity diaries [VFC10]. It uses non-geographical space-time cubes. The cube’s two data axes can be mapped to data dimensions such as individuals, locations, or activities. We focus on the case where one axis maps to individuals while the other axis maps to activities. Activities are also encoded using color.



**Figure 31:** VISUAL TimePacTS [VFC10]. (a) Space flattening on activities, (b) Oblique flattening, (c) Space flattening on individuals.

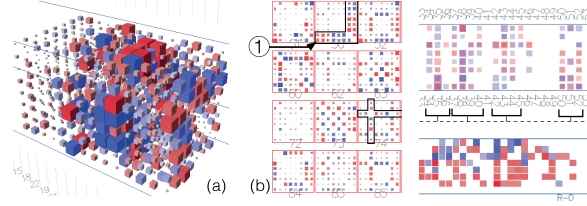
VISUAL-TimePacTS supports *linear space flattening* on both data axes. Figure 31(a) shows 6 individuals (horizontal axis) and their activities (colors) across time (vertical axis). Figure 31(c) shows the evolution of activities aggregated across all people over time. VISUAL-TimePacTS supports a seamless transition between the two operations through *interactive 3D rendering* (Figure 31(c)). Since 3D rendering employs orthographic projection and no shading, it is essentially an *oblique flattening* operation.

VISUAL-TimePacTS supports a more elaborate space-time cube operation that prevents visual marks from overlapping due to *flattening*. In Figure 31(c), for example, individuals are horizontally offset when several of them do the same activity at the same time. This technique is called *shearing* by the authors, and is further explained in Figures 31(d), 31(e). This technique is essentially a (*linear space cutting* + *space shifting*) \* *space flattening* operation, and is a hybrid between *space juxtaposing* and *space flattening*.

#### 6.4. Cubix

Cubix is a system for analyzing dynamic weighted networks through adjacency matrix representations [BPF14b]. A *3D rendering* provides an overview of the data (Figure 32(a)). Time goes from left to right. Each cell of the cube represents a connection between two nodes at a given time, with size depending on connection weight. Cells can be color-coded according to time, weight, or direction.

Cubix supports a range of space-time cube operations, including *time juxtaposing* (Figure 32(b)), *space juxtaposing* (detail in Figure 32(d)), *animated time cutting*, *animated*

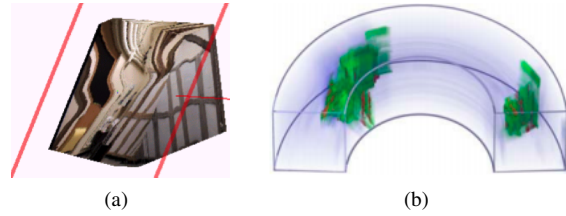


**Figure 32:** Different operations applied to a time-evolving adjacency matrix in Cubix [BPF14b]

*space cutting*, *time flattening* and *space flattening*. For flattening operations, cells can be made translucent to visually aggregate the history of connections. Cubix also supports *semantic filtering* on connections based on their weight.

Cubix provides a control widget in the form of a stylized cube, and whose different parts can be clicked or dragged to switch between operations. All operation switches are explained using animated transitions through rotations of the 3D rendering representation, or through staged animations of extraction and rigid transformation operations.

#### 6.5. Video Cube Systems



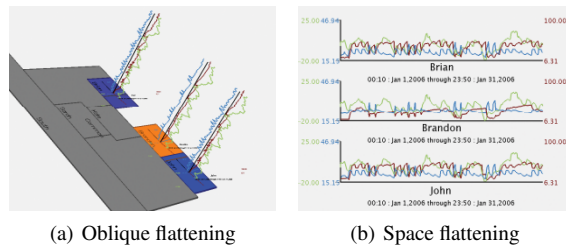
**Figure 33:** (a) Video Cubism [FLM00]; (b)  $V^3$  [DC03].

Several space-time cube systems have been proposed to support video analysis [MB98, FLM00, DC03, CI05]. Video Cubism [FLM00] uses a *3D rendering* representation together with an *interactive volume extraction* operation that is defined by manipulating a planar cutting plane (Figure 33(a)). Similarly, Khronos projector [CI05] supports manipulation of a non-planar cutting plane using touch or mid-air gestures (Figure 5).  $V^3$  [DC03] (Volume Visualization for Videos) supports different operations, including *time juxtaposing* and a *3D rendering* view that can be combined with a *bending* operation (Figure 33(b)).  $V^3$  also supports *filtering* operations that allow removal of pixels of a certain color, or pixels that do not change across a given time period.

#### 6.6. Wakame

Wakame [FW10] is a system for visualizing multivariate data by extruding starplots along a time dimension. The base plane of the space-time cube lays out starplots by their geographical location, although any layout could be imagined (Figure 34(a)). Colors correspond to dimension in the data.





**Figure 34:** Wakame [FW10]

Wakame supports 3 operations: *oblique flattening* (Figure 34(a)), *time flattening* (Figure 34(b)), and *time cutting*. In oblique flattening, extruded starplots can be rendered as traditional timeline charts as shown in Figure 34(b). The system provides an animated transition to the time-flattening where those timeline charts are shown aligned and in 2D. For time-cutting, the user selects a single time point on the complementary timeline and the system moves the camera orthogonal to the base plane.

Besides the space-time cube systems reviewed in this section, there is a wealth of general 3D visualization systems. Commercial and research tools exist in domains such as geo-visualization (e.g., Voxler [Vox], ArcGIS [arc]), scientific visualization (e.g., VTK [SAH00], Matlab [mat] and R [r]), and medical visualization [MTB03]. Although these tools do not treat time as a specific dimension, they can be used to inspire the design of interactive space-time cube systems.

## 7. General Discussion

We now discuss the limitations of our descriptive framework and consider areas for future research, including: unifying our framework with the infovis pipeline model, extending it to other dimensionalities, discussing OLAP-cubes, considering non-planar media (e.g. physical visualizations), as well as discussing user tasks.

### 7.1. Comparison with the Infovis Pipeline Model

Since our framework builds on the notion of sequential composition of operations, it shares similarities with another common model: the infovis reference model, also called the *infovis pipeline* [CMS99, Chi00, JD13]. The infovis pipeline sees visualization as a data-flow process, i.e., a sequence of stages and transformations that turn raw data into a final image. These transformations commonly include *data transformation*, *visual mapping*, *presentation mapping* and *rendering* [JD13]. Interactivity is implemented by having data analysts alter these transformations at different stages.

There is clearly an analogy between transformations on the infovis pipeline and operations on a space-time cube, since both describe steps in creating a visualization. However, the infovis pipeline describes steps for the *implementation* of a visualization, i.e. when and in which order data

is transformed and mapped onto visual variables in a visualization. Space-time cube operations, on the other hand, are purely conceptual and do not imply any direct implementation; a slice from a cube could be seen as a data object, a graphical object, something in-between, or something blending the two. Our model rather turns a 3D conceptual space into a 2D visualization space, regardless of how the visualization is eventually implemented on the infovis pipeline.

In fact, some space-time operations can be performed at different stages of the infovis pipeline, for example, *time flattening* can be done at the *data transformation* stage, by aggregating raw data over time. Alternatively, it could be emulated by explicitly rendering a 3D space-time cube on the screen and using a proper camera placement and projection transformation. In that case, it would be implemented at the *rendering* level. Frame buffers objects are another example as they can store data, while running on the GPU [ME09]. Both models, the infovis pipeline and the generalized space-time cube, are complementary.

Finally, the infovis pipeline is a *general* model for visualizations, where the sequence of operations is *fixed*, but the operations themselves are rather *abstract*. For example, the infovis pipeline model provides no specific details about what happens in the *visual mapping* transformation. In contrast, our model only captures a *specific* family of visualizations (temporal visualizations), its sequence of operations is *not fixed*, and the operations are more *concrete*. The infovis pipeline is more general but too high-level to capture the similarities and differences between the visualizations we presented. On the other hand, our model is incomplete in that it does not define how the space-time cube is built.

To fully support a variety of space-time cube operations on the same dataset, the generalized space-time cube must be implemented as a first-class object, as done in many space-time cube systems (Section 6). Thus space-time cube operations are best seen as *presentation mapping* or *rendering* transformations, i.e., transformations that turn the abstract visual form into a fully-specified 2D image or 3D model [JD13]. In other terms, our space-time cube operations can be used to decompose and refine the *presentation mapping* transformation of the infovis pipeline. We believe that implementing our framework in this way, and potentially based on abstract inner structures of space-time cubes, could dramatically facilitate the exploration of a wide range of temporal data sets.

### 7.2. Other Dimensionalities

This article focused on temporal visualizations that involve *two dimensions in the base plane* plus time. These two dimensions can be inherently spatial or can result from 2D spatial encodings of non-spatial data. However, temporal visualizations with other dimensionalities exist or may be required for specific problems.

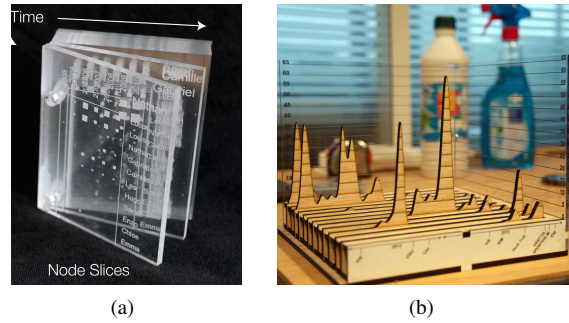
Most notably, a rich variety of temporal visualizations exist that involve a *single* spatial dimension plus time, e.g., timelines and time-series visualizations [AMST11]. In principle, our framework still applies if the 3D space-time cube is turned into a 2D space-time plane. Operations analogous to our *geometry transformation* operations would capture techniques such as spiral visualizations, calendar visualizations or cycle plots [AMST11]. However, since a 2D space-time plane already naturally maps to a 2D planar display, and since the richness of time-series visualizations and timelines mostly stem from the visual encodings used, the usefulness of our framework would be less clear in this case.

Other temporal visualization techniques, although less common, show *three* spatial dimensions plus time. We believe most of them can be captured with operations on 4-dimensional space-time hypercubes. For example, Tufte explains how small multiples can be used to show the evolution of a three-dimensional storm [BB95]. This approach amounts to applying a *time juxtaposition* operation on a space-time hypercube, where each *time cutting* operation yields a 3D image. Similarly, FromDaDy [HTC09] uses 3D trails to show the trajectories of airplanes in space. This technique amounts to performing a *time flattening* on a space-time hypercube. Extending our framework to higher data dimensionalities is an exciting topic for future research. However, it is less easy to imagine a hypercube than a cube, so the merits of such a conceptual model still remain to be seen. Another solution to explore high-dimensional temporal data is to use dimensional reduction [BSH\*16, vdeHBvW16].

### 7.3. OLAP Cubes

Related to both implementation and higher dimensionalities of the generalized space-time cube are data cubes [GCB\*97] in Online Analytical Processing (OLAP cubes). OLAP cubes consist of three data dimensions, selected freely from a potentially high dimensional dataset and do not have to involve a single temporal dimension. In this sense, OLAP cubes could be seen as superset to all space-time cubes. However, OLAP cubes are unable to capture “moving” objects such as animals or nodes in a network (Figure 26, ① to ④), hence representing only a subset of all space-time cubes.

OLAP cubes are meant for querying and aggregating multidimensional data, rather than for visualization and exploration. They are pure logical models, and we are not aware of any visualization system that explicitly uses OLAP cubes. Some OLAP operations indeed share striking analogies with STC operations, namely *slice* (*time* and *space cutting*), *dice* (*space* and *time chopping*) and *roll-up* (*space* and *time flattening*). Other operations on OLAP cubes, not explicitly supported by space-time cubes, are *drill-down* (zooming in and effectively being the reverse of *flattening*) and *pivot* (reassigning dimensions of the cube). In supporting these operations, the spatial structure of the OLAP cube can vary and is not persistent, as in space-time cubes.



**Figure 35:** Examples of physical space-time cubes: a) *Matrix Cube* for dynamic network [BPF14b] (made by Benjamin Bach), b) *Subversion commits* on different research projects over time [SSJ\*14].

Future work could extend our framework to OLAP cubes. This would include to i) differentiate between the two dimensions of the base plane, and consequently to ii) extend the number of existing operations (e.g. enable cutting on each of the three cube dimensions). Finally, operations currently supported by OLAP cubes should be discussed for their general applicability to actual space-time cubes.

### 7.4. Non-Planar Media

Throughout this article we assumed the presentation medium to be planar. Although these are by far the most common, other display shapes are being explored in HCI, some of which are even deformable [RPPH12, HV08]. In these cases, the conditions for an operation to be *complete* are not the same. This opens up a wide range of possibilities for new visualization designs. For example, one implementation of the Khronos projector (Figure 5) employs back projection on a freely deformable cloth, allowing the use of *non-planar cutting* operations that are *complete*. In addition, physical visualizations make it possible to faithfully display 3D space-time cubes without any additional operations [JDF13, JD13]. Many such visualizations have been already crafted by scientists, artists and designers [DJ13].

Physical temporal visualizations can even be made modular to support interactive space-time cube operations. Figure 35(a) shows two physical representations of a dynamic network [BPF14b] made of laser-cut and laser-engraved acrylic. The left version supports interactive *time cutting* while the right version supports interactive *space cutting*. Cuts can be taken apart and manipulated freely, allowing for *time juxtaposing* and *space juxtaposing* as well as *time flattening* and *space flattening*, if viewed from a proper orthogonal angle and distance. Figure 35(b) shows SVN commits over time. Time goes from left to right, and different layers represent different projects. Slices can also be taken apart and reordered. For another example see [STB13].

### 7.5. Towards a Prescriptive Model

Our framework is mostly descriptive. A prescriptive model in turn can help us making implications about which operations to use for which data and tasks. Unfortunately, we need more empirical studies to assess the relative effectiveness of different space-time cube operations with respect to specific tasks as well as data types and characteristics. Although many studies have already been conducted (as discussed before), the current body of evidence paints a complex and partly inconclusive picture. This may be due to the lack of a clear and detailed descriptive framework. A clear descriptive framework allows us to better tease apart the effects of different subtle design features and to better control for confounds. We showed how many temporal visualization techniques can be decomposed into elementary operations. These operations can be combined in many ways or made dynamic at different levels, either through animation or interaction. The characterization of the inner structure of space-time cubes may already provide many elements to discuss the practical strengths and weaknesses of space-time cube operations, mostly based on well-established knowledge on perception and HCI, and on common sense.

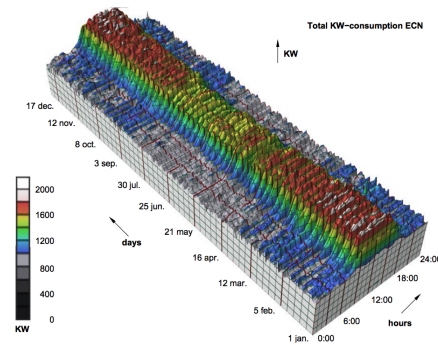
Running studies for answering specific research questions will naturally remain important, and we hope our descriptive framework will facilitate the design of such studies and the discussion of findings in a more informative manner, leading to a more prescriptive framework in the future.

### 7.6. User Tasks

In our discussion on the advantages and drawbacks of certain space-time cube operations, we have mostly focused on the inner structure of the space-time cubes as a source of trade-offs. However, the process of creating a successful visualization also has to take into account the user's tasks. There have been efforts at proposing task taxonomies that should complement our framework (e.g. [Mac95, AA05, Rot13, APS14, BPF14a]). However, we do not yet see a straightforward mapping between operations and the tasks they support best. As already mentioned, more effort is needed to come up with domain-agnostic task taxonomies.

### 7.7. Other Limitations

Our framework is only meant to be a thinking tool. Like any model, it is necessarily incomplete. First, our taxonomy of elementary operations in Figure 24 is not meant to cover all possible operations since their number is potentially infinite and the taxonomy can easily be extended. Second, our framework does not provide much guidance for interaction design: the design space for interactive operations has only been partially explored in research. Finally, not all techniques for visualizing temporal data can be captured with space-time cubes. For example, temporal data can be visualized using two time axes instead of a single data axis



**Figure 36:** Energy consumption over time [vWvS99]: the horizontal axis is mapped to days while the other one is mapped to hours.

(see Figure 36). Also, space-time cubes may not account for branching times. However, our framework is not meant to restrict creativity but rather to help visualization designers find new solutions, extend or generalize existing ones, and think out of the box.

Our framework assumes that the space-time cube already exists. It does not provide guidance for producing the space-time cube itself, mainly because this process involves many decisions and should not restrict the range of possible techniques. For non-spatial data, many visual mappings can be used to produce individual slices, for example, locations on a map can yield values for altitude, temperature, rain, vegetation and soil type. How to visualize all these attributes at any particular point in time is a general problem of information visualization, but the choice may also affect the efficiency of later space-time cube operations. We hope that future work can clarify some of these questions.

### 8. Conclusion

We reviewed various techniques to visualize temporal data, by describing them as sequences of parametric operations applied to a conceptual space-time cube. Our operations are independent from the underlying data and can be applied across a range of application domains, be they cartography, dynamic network analysis, geopolitics, or video analytics. When choosing the appropriate operations, a designer only needs to take into account the inner structure of the space-time cubes generated by the data, and the users' tasks.

Furthermore, by introducing domain-agnostic concepts and a clear terminology, this article aims at facilitating the comparison of different approaches for visualizing temporal data. Existing visualizations from one data domain can be analyzed in terms of elementary operations and then be adapted to other domains and problems. By giving a better vision of the richness of this design space, we hope our model will also motivate the exploration of new approaches. It stresses the importance of developing fully-integrated interactive systems and toolkits that can support a range of techniques in a consistent manner.



Our model further aims at facilitating the design of studies and discussing their results in a more informative as well as formal manner. We hope the presented terminology and low-level concepts will help design better experiments that tease out important factors in dynamic data visualization. Controlled studies are needed to understand the trade-offs between different space-time cube operations and how they perform depending on task, data, and the people using them.

This work mostly arose out of the need to teach temporal information visualization to undergrad students. We therefore hope that it will help other people teach this field effectively, by providing a clear structure and a clear terminology on which to base higher-level discussions and analyses.

### Acknowledgements

We would like to thank the anonymous reviewers, as well as the presentation audience at EuroVis 2014 for their valuable feedback. Collaboration on this work was initiated by Dagstuhl Seminar “Putting Data on the Map 12261”, June 2012, and supported in part by NSERC, SMART Technologies, AITF, Surfnet and GRAND NCE, as well as by Clique Strategic Research Cluster funded by Science Foundation Ireland (SFI) Grant No. 08/SRC/I1407.

### References

- [AA04] ANDRIENKO N., ANDRIENKO G.: Interactive visual tools to explore spatio-temporal variation. In *Proc. of the Working Conference on Advanced Visual Interfaces (AVI)* (2004), ACM, pp. 417–420. 1, 2, 7
- [AA05] ANDRIENKO N., ANDRIENKO G.: *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer-Verlag, Berlin Heidelberg, Germany, 2005. 22
- [AA12] ANDRIENKO N., ANDRIENKO G.: Visual analytics of movement: An overview of methods, tools and procedures. *Information Visualization* 12, 1 (2012), 3–24. 2
- [AAB\*11] ANDRIENKO G., ANDRIENKO N., BAK P., KEIM D., KISILEVICH S., WROBEL S.: A Conceptual Framework and Taxonomy of Techniques for Analyzing Movement. *Journal of Visual Languages and Computing* 22, 3 (June 2011), 213–232. 2
- [AAH11] ANDRIENKO G., ANDRIENKO N., HEURICH M.: An Event-based Conceptual Model for Context-aware Movement Analysis. *International Journal on Geographic Information Science* 25, 9 (Sept. 2011), 1347–1370. 2
- [AMM\*07] AIGNER W., MIKSCH S., MÜLLER W., SCHUMANN H., TOMINSKI C.: Visualizing Time-oriented data-A Systematic View. *Computers and Graphics* 31, 3 (June 2007), 401–409. 2
- [AMST11] AIGNER W., MIKSCH S., SCHUMANN H., TOMINSKI C.: *Visualization of Time-Oriented Data*. Human-Computer Interaction. Springer, London, UK, 2011. 1, 21
- [AP12a] ARCHAMBAULT D., PURCHASE H. C.: Mental Map Preservation Helps User Orientation in Dynamic Graphs. In *Proc. of Graph Drawing* (2010), Springer, pp. 475–486. 10, 17
- [AP12b] ARCHAMBAULT D., PURCHASE H. C.: The Mental Map and Memorability in Dynamic Graphs. In *Proc. of Pacific Visualization Symposium* (2012), IEEE, pp. 89–96. 17
- [AP13] ARCHAMBAULT D., PURCHASE H. C.: The “Map” in the Mental Map: Experimental Results in Dynamic Graph Drawing. *International Journal of Human-Computer Studies* 71, 11 (2013), 1044–1055. 16
- [AP16] ARCHAMBAULT D., PURCHASE H. C.: Can animation support the visualisation of dynamic graphs? *Information Sciences* 330 (2016), 495–509. 17
- [APP10] ARCHAMBAULT D., PURCHASE H. C., PINAUD B.: Difference map readability for dynamic graphs. In *Proc. of Graph Drawing* (2010), vol. 6502, Springer, pp. 50–61. 16
- [APP11] ARCHAMBAULT D., PURCHASE H., PINAUD B.: Animation, Small Multiples, and the Effect of Mental Map Preservation in Dynamic Graphs. *IEEE Transactions on Visualization and Computer Graphics* 17, 4 (2011), 539–552. 17
- [APS14] AHN J.-W., PLAISANT C., SHNEIDERMAN B.: A task taxonomy for network evolution analysis. *IEEE Transactions on Visualization and Computer Graphics* 20, 3 (Mar. 2014), 365–376. 22
- [arc] ArcGIS. <http://www.esri.com/software/arcgis>. [online, accessed:02-apr-2014]. 20
- [AS] ARIS A., SHNEIDERMAN B.: NVSS: Network Visualization by Semantic Substrates. <http://www.cs.umd.edu/hcil/nvss/>. [online, accessed:02-apr-2014]. 6
- [ATMS\*11] AHN J.-W., TAIEB-MAIMON M., SOPAN A., PLAISANT C., SHNEIDERMAN B.: Temporal visualization of social network dynamics: prototypes for nation of neighbors. In *Proc. of International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction* (2011), Springer, pp. 309–316. 11
- [Bar97] BARTRAM L.: Perceptual and interpretative properties of motion for information visualization. In *Proc. of Workshop on New paradigms in Information Visualization and Manipulation* (1997), ACM, pp. 3–7. 16
- [BB95] BAKER M. P., BUSHELL C.: After the storm: Considerations for information visualization. *IEEE Computer Graphics and Applications* 15, 3 (1995), 12–15. 21
- [BBL12] BOYANDIN I., BERTINI E., LALANNE D.: A Qualitative Study on the Exploration of Temporal Changes in Flow Maps with Animation and Small-Multiples. *Computer Graphics Forum* 31, 3pt2 (2012), 1005–1014. 5
- [BC87] BECKER R. A., CLEVELAND W. S.: Brushing Scatterplots. *Technometrics* 29, 2 (1987), 127–142. 15
- [BC03] BRANDES U., CORMAN S. R.: Visual unrolling of network evolution and the analysis of dynamic discourse. *Information Visualization* 2, 1 (Mar. 2003), 40–50. 7, 13
- [BCD\*12] BORGO R., CHEN M., DAUBNEY B., GRUNDY E., HEIDEMANN G., HÖFERLIN B., HÖFERLIN M., LEITTE H., WEISKOPF D., XIE X.: State of the Art Report on Video-Based Graphics and Video Visualization. *Computer Graphics Forum* 31, 8 (Dec. 2012), 2450–2477. 2
- [BCH07] BRUNSDON C., CORCORAN J., HIGGS G.: Visualising space and time in crime patterns: A comparison of methods. *Computers, Environment and Urban Systems* 31, 1 (2007), 52–75. Extracting Information from Spatial Datasets. 17
- [BDA\*14] BACH B., DRAGICEVIC P., ARCHAMBAULT D., HURTER C., CARPENDALE S.: A Review of Temporal Data Visualizations Based on Space-Time Cube Operations. In *Proc. of Eurographics Conference on Visualization (EuroVis)* (Aire-la-Ville, Switzerland, June 2014). 1
- [BDH04] BARTOLI A., DALAL N., HORAUD R.: Motion Panoramas. *Computer Animation and Virtual Worlds*, 15 (2004), 501–517. 5
- [BGPS07] BATTIATO S., GALLO G., PUGLISI G., SCELLATO S.: SIFT features tracking for video stabilization. In *Proc. of Conference on Image Analysis and Processing* (2007), IEEE, pp. 825–830. 10

- [BHRD\*15] BACH B., HENRY-RICHE N., DWYER T., MADHYASTHA T., FEKETE J.-D., GRABOWSKI T.: Small MultiPiles: Piling Time to Explore Temporal Patterns in Dynamic Networks. *Computer Graphics Forum* (2015), 559–568. 5, 10
- [BKH\*16] BACH B., KERRACHER N., HALL K. W., CARPENDALE S., KENNEDY J., RICHE N. H.: Telling stories about dynamic networks with graph comics. In *Proc. of ACM Conference on Human Factors in Information Systems (CHI)* (2016), ACM, to appear. 5
- [BN11] BRANDES U., NICK B.: Asymmetric Relations in Longitudinal Social Networks. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (Dec. 2011), 2283–2290. 7
- [BPF14a] BACH B., PIETRIGA E., FEKETE J.-D.: GraphDiaries: Animated Transitions and Temporal Navigation for Dynamic Networks. *IEEE Transactions on Visualization and Computer Graphics* 20, 5 (2014), 740 – 754. 5, 10, 11, 16, 22
- [BPF14b] BACH B., PIETRIGA E., FEKETE J.-D.: Visualizing Dynamic Networks with Matrix Cubes. In *Proc. of ACM Conference on Human Factors and Computing Systems (CHI)* (2014), ACM. 2, 9, 10, 11, 14, 19, 21
- [BSH\*16] BACH B., SHI C., HEULOT N., MADHYASTHA T., GRABOWSKI T., DRAGICEVIC P.: Time Curves: Folding Time to Visualize Patterns of Temporal Evolution in Data. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (Jan. 2016). 21
- [BVB\*11] BURCH M., VEHLW C., BECK F., DIEHL S., WEISKOPF D.: Parallel Edge Splatting for Scalable Dynamic Graph Visualization. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2344–2353. 6
- [CAS05] CONTI G., AHAMAD M., STASKO J.: Attacking information visualization system usability overloading and deceiving the human. In *Proc. of Symposium on Usable Privacy and Security* (2005), ACM, pp. 89–100. 14
- [CCF96] COWPERTHWAIT D. J., CARPENDALE M. S. T., FRACCHIA F. D.: Visual access for 3D data. In *Proc. of ACM Conference on Human Factors in Information Systems (CHI)* (1996), ACM, pp. 175–176. 14
- [CCT\*99] CARPENDALE S. T., COWPERTHWAIT D. J., TIGGES M., FALL A., FRACCHIA F. D.: The Tardis: A Visual Exploration Environment for Landscape Dynamics. In *Proc. of Conference on Visual Data Exploration and Analysis* (1999), no. 3643, pp. 110–119. 2, 9, 10, 12, 18
- [CFC\*96] CARPENDALE S. T., FALL A., COWPERTHWAIT D. J., FALL J., FRACCHIA F. D.: Case study: visual access for landscape event based temporal data. In *Proc. of IEEE Visualization* (1996), IEEE, pp. 425–428. 14, 18
- [Chi00] CHI E. H.-H.: A Taxonomy of Visualization Techniques using the Data State Reference Model. In *Proc. of IEEE Symposium on Information Visualization* (2000), IEEE, pp. 69–75. 20
- [CI05] CASSINELLI A., ISHIKAWA M.: Khronos Projector. In *Proc. of Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (2005), ACM. 2, 19
- [CKN\*03] COLLBERG C., KOBOUROV S., NAGRA J., PITTS J., WAMPLER K.: A System for Graph-Based Visualization of the Evolution of Software. In *Proc. of ACM Symposium on Software Visualization (SoftVis)* (2003), ACM, pp. 77–ff. 5
- [CMS99] CARD S. K., MACKINLAY J. D., SHNEIDERMAN B. (Eds.): *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1999. 20
- [DC03] DANIEL G., CHEN M.: Video Visualization. In *Proc. of IEEE Visualization* (2003), IEEE, pp. 409–416. 10, 19
- [DE02] DWYER T., EADES P.: Visualising a Fund Manager Flow Graph with Columns and Worms. In *Proc. of International Conference on Information Visualisation* (Los Alamitos, CA, 2002), IEEE, pp. 147–152. 7
- [DG04] DWYER T., GALLAGHER D. R.: Visualising changes in fund manager holdings in two and a half-dimensions. *Information Visualization* 3, 4 (Dec. 2004), 227–244. 7
- [DJ13] DRAGICEVIC P., JANSEN Y.: List of Physical Visualizations. <http://www.tinyURL.com/physvis>, 2013. [Online; accessed 04-Sep-2013]. 21
- [DV10] DEMŠAR U., VIRRANTAUŠ K.: Space-time density of trajectories: exploring spatio-temporal patterns in movement data. *International Journal of Geographical Information Science* 24, 10 (2010), 1527–1542. 10, 14
- [ELMS] EADES P., LAI W., MISUE K., SUGIYAMA K.: Preserving the mental map of a diagram. In *Proc. of Computer Graphics*, International Institute for Advanced Study of Social Information Science, Fujitsu Limited. 16
- [FBS06] FALKOWSKI T., BARTELHEIMER B., SPILIOPOULOU M.: Mining and Visualizing the Evolution of Subgroups in Social Networks. In *Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on* (2006), pp. 52–58. 6
- [FLM00] FELS S., LEE E., MASE K.: Techniques for Interactive Video Cubism (poster session). In *Proc. of International Conference on Multimedia* (2000), ACM, pp. 368–370. 2, 9, 19
- [FQ11] FARRUGIA M., QUIGLEY A.: Effective temporal graph layout: A comparative study of animations versus static display methods. *Information Visualization* 10, 1 (2011), 47–64. 17
- [FT06] FELDMAN J., TREMOULET P. D.: Individuation of visual objects over time. *Cognition* 99, 2 (2006), 131–165. 16
- [Fur86] FURNAS G. W.: Generalized fisheye views. In *Proc. of the ACM Conference on Human Factors in Computing Systems (CHI)* (1986), ACM, pp. 16–23. 3
- [FW10] FORLINES C., WITTENBURG K.: Wakame: Sense making of multi-dimensional spatial-temporal data. In *Proc. of the International Conference on Advanced Visual Interfaces (AVI)* (2010), ACM, pp. 33–40. 19, 20
- [GAA04] GATASKY P., ANDRIENKO N., ANDRIENKO G.: Interactive Analysis of Event Data Using Space-Time Cube. In *Proc. of the Information Visualisation* (2004), IEEE, pp. 145–152. 7
- [GCB\*97] GRAY J., CHAUDHURI S., BOSWORTH A., LAYMAN A., REICHAUT D., VENKATRAO M., PELLOW F., PIRAHESH H.: Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining in Knowledge Discovery* 1, 1 (Jan. 1997), 29–53. 21
- [geo] GeoTime. <http://www.geotime.com>. [Online; accessed 24-Jan-2014]. 18
- [GEY12] GHANI S., ELMQVIST N., YI J. S.: Perception of Animated Node-Link Diagrams for Dynord University Pressamic Graphs. In *Computer Graphics Forum* (2012), The Eurographics Association and Blackwell Publishing Ltd. 17
- [GHW09] GROH G., HANSTEIN H., WOERNDL W.: Interactively Visualizing Dynamic Social Networks with DySoN. In *Workshop on Visual Interfaces to the Social and the Semantic Web* (February 2009), ACM. 7
- [GMH\*06] GRIFFEN A. L., MACEachren A. M., HARDISTY F., STEINER E., LI B.: A Comparison of Animated Maps with Static Small-Multiple Maps for Visually Identifying Space-Time Clusters. *Annals of the Association of American Geographers* 96, 4 (2006), 740–753. 16, 17

- [Gre11] GRETCHEN P.: A Cartographer's Toolkit - Small Multiples. <http://www.gretchenpeterson.com/blog/small-multiples>, 2011. [Online; accessed 24-Jan-2014]. 5
- [H70] HÄGERSTRAND T.: What about People in Regional Science. In *Papers of the Regional Science Association* (1970). 2
- [Har99] HARRIS R. L.: *Information Graphics: A Comprehensive Illustrated Reference*. Oxford University Press, New York, NY, 1999. 1
- [HEF\*13] HURTER C., ERSOY O., FABRIKANT S., KLEIN T., TELEA A.: Bundled Visualization of Dynamic Graph and Trail Data. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 20 (2013), 1141–1157. 10, 11
- [HET12] HURTER C., ERSOY O., TELEA A.: Graph Bundling by Kernel Density Estimation. *Computer Graphics Forum* 31 (June 2012), 865–874. 11
- [HKV12] HU Y., KOBOUROV S. G., VEERAMONI S.: Embedding, Clustering and Coloring for Dynamic Maps. In *Proc. of Pacific Visualization Symposium* (2012), PacificVis '12, IEEE. 10
- [HSCW13] HADLAK S., SCHUMANN H., CAP C. H., WOLLENBERG T.: Supporting the Visual Analysis of Dynamic Networks by Clustering associated Temporal Attributes. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 19, 12 (2013), 2267–2276. 7
- [HTC09] HURTER C., TISSOIRE B., CONVERSY S.: From-DaDy: Spreading Aircraft Trajectories Across Views to Support Iterative Queries. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1017–1024. 21
- [HV08] HOLMAN D., VERTEGAAL R.: Organic user interfaces: designing computers in any way, shape, or form. *Communications of the ACM* 51, 6 (2008), 48–55. 21
- [JD13] JANSEN Y., DRAGICEVIC P.: An interaction Model for Visualizations Beyond The Desktop. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 19, 12 (2013), 2396–2405. 7, 20, 21
- [JDF13] JANSEN Y., DRAGICEVIC P., FEKETE J.-D.: Evaluating the Efficiency of Physical Visualizations. In *Proc. of the ACM Conference on Human Factors in Computing Systems (CHI)* (2013), ACM, pp. 2593–2602. 21
- [KDA\*09] KRISTENSSON P., DAHLBACK N., ANUNDI D., BJORNSTAD M., GILLBERG H., HARALDSSON J., MARTENSSON I., NORDVALL M., STAHL J.: An Evaluation of Space Time Cube Representation of Spatiotemporal Patterns. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 15, 4 (July 2009), 696–702. 17
- [KK12] KVELADZE I., KRAAK M.-J.: What do we know about the space-time cube from cartographic and usability perspective? In *Proc. of the 2012 AutoCarto International Symposium on Automated Cartography* (2012). 17
- [KPW13] KERREN A., PURCHASE H. C., WARD M. O. (Eds.): *Multivariate Network Visualization*, vol. 8380. Springer, Berlin Heidelberg, Germany, 2013. 13
- [Kra03] KRAAK M. J.: The Space-Time Cube revisited from a Geovisualization Perspective. *Proc. of the International Cartographic Conference* (2003), 1988–1996. 2, 7
- [KW04] KAPLER T., WRIGHT W.: GeoTime Information Visualization. In *Proc. of IEEE Information Visualization* (2004), INFOVIS '04, IEEE, pp. 25–32. 9, 14, 18
- [LNS11] LIU Z., NAVATHE S., STASKO J.: Network-based visual analysis of tabular data. In *Proc. of Conference on Visual Analytics Science and Technology* (2011), IEEE, pp. 41–50. 5
- [Mac95] MACEACHREN A. M.: *How Maps Work*. Guilford Press, 1995. 22
- [Mar78] MAREY É.-J.: *La méthode graphique dans les sciences expérimentales et particulièrement en physiologie et en médecine*. G. Masson, 1878. 4, 6
- [mat] MatLab. <http://www.mathworks.fr/products/matlab/>. [online, accessed:02-apr-2014]. 20
- [MB98] MACKAY W., BEAUDOUIN-LAFON M.: DIVA: Exploratory Data Analysis with Multimedia Streams. In *Proc. of ACM Conference on Human Factors in Computing Systems (CHI)* (apr 1998), ACM, pp. 416–423. 19
- [McC94] MCCLOUD S.: *Understanding Comics*. HarperCollins, New York, NY, 1994. 5, 15
- [ME09] McDONNELL B., ELMQVIST N.: Towards Utilizing GPUs in Information Visualization: A Model and Implementation of Image-Space Operations. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1105–1112. 20
- [MELS95] MISUE K., EADES P., LAI W., SUGIYAMA K.: Layout Adjustment and the Mental Map. *Journal of Visual Languages and Computing* 6 (1995), 183–210. 16
- [MR97] MATTHEWS G., ROZE M.: Worm plots. *Computer Graphics and Applications* 17, 6 (Nov 1997), 17–20. 7
- [MTB03] MCGUFFIN M. J., TANCAU L., BALAKRISHNAN R.: Using Deformations for Browsing Volumetric Data. In *Proc. of IEEE Visualization* (October 2003), IEEE, pp. 401–408. 20
- [Mun15] MUNZNER T.: *Visualization Analysis & Design*. CRC Press, Boca Raton, FL, 2015. 13, 16
- [Muy87] MUYBRIDGE E.: *Animal Locomotion*. Published under the auspices of the University of Pennsylvania, Philadelphia, PA, 1887. 5
- [PHG06] PURCHASE H. C., HOGGAN E., GÖRG C.: How important is the “Mental Map”? – An Empirical Investigation of a Dynamic Graph Layout Algorithm. In *Proc. of Graph Drawing* (2006), vol. 4372, Springer, pp. 184–195. 17
- [PJH05] PETROVIC N., JOJIC N., HUANG T. S.: Adaptive Video Fast Forward. *Multimedia Tools and Applications* 26, 3 (2005), 327–344. 10, 16
- [PS08] PURCHASE H. C., SAMRA A.: Extremes Are Better: Investigating Mental Map Preservation in Dynamic Graphs. In *Proc. of Diagrams* (2008), vol. 5223, Springer, pp. 60–73. 17
- [r] R. <http://www.r-project.org/>. [online, accessed:02-apr-2014]. 20
- [Ren02] RENSINK R. A.: Change Detection. *Annual Review of Psychology* 53, 1 (2002), 245–277. 16
- [RFF\*08] ROBERTSON G., FERNANDEZ R., FISHER D., LEE B., STASKO J.: Effectiveness of Animation in Trend Visualization. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (Nov. 2008), 1325–1332. 9, 10, 11, 17
- [RM13] RUFIANGE S., MCGUFFIN M. J.: DiffAni: Visualizing Dynamic Graphs with a Hybrid of Difference Maps and Animation. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2556–2565. 5, 10, 11, 16
- [Ros06] ROSLING H.: Stats that reshape your worldview. [http://www.ted.com/talks/hans\\_rosling\\_the\\_good\\_news\\_of\\_the\\_decade.html](http://www.ted.com/talks/hans_rosling_the_good_news_of_the_decade.html), 2006. [online, accessed:24-jan-2014]. 3, 10, 11
- [Rot13] ROTH R.: An empirically-derived taxonomy of interaction primitives for interactive cartography and geovisualization. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013). 22



- [RPPH12] RASMUSSEN M. K., PEDERSEN E. W., PETERSEN M. G., HORNBAEK K.: Shape-changing interfaces: a review of the design space and open research questions. In *Proc. of the ACM Conference on Human Factors in Computing Systems (CHI)* (2012), ACM, pp. 735–744. 21
- [SA06] SHNEIDERMAN B., ARIS A.: Network Visualization by Semantic Substrates. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 12, 5 (2006), 733–740. 6
- [SAH00] SCHROEDER W. J., AVILA L. S., HOFFMAN W.: Visualizing with VTK: a tutorial. *IEEE Computer Graphics and Applications* 20, 5 (2000), 20–27. 20
- [Shn03] SHNEIDERMAN B.: Why not make Interfaces better than 3D Reality? *IEEE Computer Graphics and Applications* 23, 6 (2003), 12–15. 13, 14
- [SP08] SAFFREY P., PURCHASE H. C.: The “Mental Map” versus “Static Aesthetic” Compromise in Dynamic Graphs: A User Study. In *Proc. of Australasian User Interface* (2008), vol. 76, Australian Computer Society, pp. 85–93. 17
- [SRdJ05] SHANBHAG P., RHEINGANS P., DES JARDINS M.: Temporal Visualization of Planning Polygons for Efficient Partitioning of Geo-Spatial Data. In *Proc. of the 2005 IEEE Symposium on Information Visualization* (Los Alamitos, CA, 2005), IEEE, pp. 211–218. 1, 2, 7
- [SSJ\*14] SWAMINATHAN S., SHI C., JANSEN Y., DRAGICEVIC P., OEHLBERG L. A., FEKETE J.-D.: Supporting the design and fabrication of physical visualizations. In *Proc. of the ACM Conference on Human Factors in Computing Systems (CHI)* (2014), ACM, pp. 3845–3854. 21
- [SSS\*01] SMITH G., STUERZLINGER W., SALZMAN T., WATSON B., BUCHANAN J.: 3D Scene Manipulation with 2D Devices and Constraints. In *Graphics Interface* (2001), vol. 1, Canadian Information Processing Society, pp. 135–142. 14
- [STB13] STUSAK S., TABARD A., BUTZ A.: Can physical visualizations support analytical tasks? *Proc. of IEEE VIS*, poster abstracts. 21
- [TBC13] THUDT A., BAUR D., CARPENDALE S.: Visits: A Spatiotemporal Visualization of Location Histories. In *EuroVis - Short Papers* (2013), The Eurographics Association. 5, 11
- [TGF08] TANG A., GREENBERG S., FELS S.: Exploring Video Streams using Slit-Tear Visualizations. In *Proc. of the working conference on Advanced visual interfaces (AVI)* (2008), ACM, pp. 191–198. 6
- [TKB07] TURDUKULOV U. D., KRAAK M.-J., BLOK C. A.: Designing a visual environment for exploration of time series of remote sensing data: In search for convective clouds. *Computers & Graphics* 31, 3 (2007), 370–379. 14
- [TMB02] TVERSKY B., MORRISON J., BETRANCOURT M.: Animation: Can it facilitate? *International Journal of Human-Computer Studies* 57, 4 (2002), 247–262. 16, 17
- [Tod04] TODD J. T.: The Visual Perception of 3D Shape. *Trends in Cognitive Sciences* 8, 3 (2004), 115–121. 14
- [TR09] THAKUR S., RHYNE T.-M.: Data vases: 2d and 3d plots for visualizing multiple time series. In *Advances in Visual Computing*, Bebis G., Boyle R., Parvin B., Koracin D., Kuno Y., Wang J., Pajarola R., Lindstrom P., Hinkenjann A., Encarnação M., Silva C., Coming D., (Eds.), vol. 5876. Springer, Berlin Heidelberg, Germany, 2009, pp. 929–938. 7
- [TS12] TOMINSKI C., SCHULZ H.-J.: The Great Wall of Space-Time. *Vision, Modeling and Visualization* (2012). 9, 12
- [TSWS05] TOMINSKI C., SCHULZE-WOLLGAST P., SCHUMANN H.: 3D information visualization for time dependent data on maps. In *Proc. of IEEE Information Visualisation* (July 2005), IEEE, pp. 175–181. 7
- [Tuf] TUFTE E.: Megan Jaegerman’s brilliant news graphics. [http://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg\\_id=0002w4](http://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=0002w4). [online, accessed:12-apr-2014]. 5
- [Tuf86] TUFTE E. R.: *The visual display of quantitative information*. Graphics Press, Cheshire, CT, 1986. 1, 4, 5, 6
- [TV07] TRUONG B. T., VENKATESH S.: Video Abstraction: A systematic Review and Classification. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)* 3, 1 (2007), 3. 5
- [vdEBvW13] VAN DEN ELZEN S., HOLTEN D., BLAAS J., VAN WIJK J.: Dynamic Network Visualization with Extended Massive Sequence Views. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* PP, 99 (2013), 1–1. 15
- [vdEBvW16] VAN DEN ELZEN S., HOLTEN D., BLAAS J., VAN WIJK J.: Reducing Snapshots to Points: A Visual Analytics Approach to Dynamic Network Exploration. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 22, 1 (2016), 1–10. 21
- [VFC10] VROTSOU K., FORSELL C., COOPER M.: 2D and 3D representations for feature recognition in time geographical diary data. *Information Visualization* 9, 4 (Dec. 2010), 263–276. 18, 19
- [Vox] Voxler. <http://www.goldensoftware.com/products/voxler>. [online, accessed:02-apr-2014]. 20
- [VWD04] VIÉGAS F. B., WATTENBERG M., DAVE K.: Studying cooperation and conflict between authors with history flow visualizations. In *Proc. of ACM Conference on Human Factors in Computing Systems (CHI)* (2004), ACM, pp. 575–582. 6, 10
- [vWvS99] VAN WIJK J. J., VAN SELOW E. R.: Cluster and Calendar Based Visualization of Time Series Data. In *Proc. of IEEE Symposium on Information Visualization* (1999), INFOVIS ’99, IEEE, pp. 4–. 22
- [War12] WARE C.: *Information Visualization: Perception for Design*. Elsevier, Orlando, FL, 2012. 15
- [WB05] WARE C., BOBROW R.: Supporting visual queries on medium sized node-link diagrams. *Journal of Information Visualization* 4, 1 (2005), 49–58. 16, 17
- [WFG92] WANGER L. R., FERWERDA J., GREENBERG D. P.: Perceiving Spatial Relationships in Computer-generated Images. *IEEE Computer Graphics and Applications* 12, 3 (1992), 44–58. 14
- [Wik13] WIKIMEDIA: Stroke Order Project. <http://www.tinyurl.com/strokeOrderProject>, 2013. [Online; accessed 24-Jan-2014]. 5
- [WvdWvW09] WILLEMS N., VAN DE WETERING H., VAN WIJK J. J.: Visualization of Vessel Movements. In *Proc. of the Conference on Visualization* (2009), Eurographics Association, pp. 959–966. 17
- [WvdWvW11] WILLEMS N., VAN DE WETERING H., VAN WIJK J. J.: Evaluation of the visibility of vessel movement features in trajectory visualizations. In *Proc. of the Eurographics Conference on Visualization (EuroVis)* (2011), Eurographics Association, pp. 801–810. 17