Computational Methods for High-Dimensional Rotations in Data Visualization

ANDREAS BUJA[1] DIANNE COOK[2],
DANIEL ASIMOV[3], CATHERINE HURLEY[4]

March 31, 2004

There exist many methods for visualizing complex relations among variables of a multivariate dataset. For pairs of quantitative variables, the method of choice is the scatterplot. For triples of quantitative variables, the method of choice is 3-D data rotations. Such rotations let us perceive structure among three variables as shape of point scatters in virtual 3-D space.

Although not obvivous, three-dimensional data rotations can be extended to higher dimensions. The mathematical construction of high-dimensional data rotations, however, is not an intuitive generalization. Whereas three-dimensional data rotations are thought of as *rotations of an object* in space, a proper framework for their high-dimensional extension is better based on *rotations of a low-dimensional projection* in high-dimensional space. The term "data rotations" is therefore a misnomer, and something along the lines of "high-to-low dimensional data projections" would be technically more accurate.

To be useful, virtual rotations need to be under interactive user control, and they need to be animated. We therefore require projections not as static pictures but as movies under user control. Movies, however, are mathematically speaking one-parameter families of pictures. This article is therefore about *one-parameter families of low-dimensional projections in high-dimensional data spaces.*

We describe several algorithms for dynamic projections, all based on the idea of smoothly interpolating a discrete sequence of projections. The algorithms lend themselves to the implementation of interactive visual exploration tools of high-dimensional data, such as so-called grand tours, guided tours and manual tours.

[1]Statistics Department, The Wharton School, University of Pennsylvania, 471 Huntsman Hall, Philadelphia, PA 19104-6302; http://www-stat.wharton.upenn.edu/~buja/

[2]Dept of Statistics, Iowa State University, Ames, IA 50011; dicook@iastate.edu, http://www.public.iastate.edu/~dicook/

[3]Mathematics Department, University of California, Berkeley, CA 94720; asimov@msri.org.

[4]Mathematics Department, National University of Ireland, Maynooth Co. Kildare, Ireland; churley@maths.may.ie, http://www.maths.may.ie/staff/churley/churley.html

1 Introduction

Motion graphics for data analysis have long been almost synonymous with 3-D data rotations. The intuitive appeal of 3-D rotations is due to the power of human 3-D perception and the natural controls they afford. To perform 3-D rotations, one selects triples of data variables, spins the resulting 3-D pointclouds, and presents a 2-D projection thereof to the viewer of a computer screen. Human interfaces for controlling 3-D data rotations follow natural mental models: Thinking of the pointcloud as sitting inside a globe, one enables the user to rotate the globe around its north-south axis, for example, or to pull an axis into oblique viewing angles, or to push the globe into continuous motion with a sweeping motion of the hand (the mouse, that is).

The mental model behind these actions proved natural to such an extent that an often asked question became vexing and almost unanswerable: How would one generalize 3-D rotations to higher dimensions? If 3-D space presents the viewer with one hidden backdimension, the viewer of $p > 3$ dimensions would face $p - 2 > 1$ backdimensions and wouldn't know how to use them to move the p-dimensional pointcloud!

Related is the fact that in 3-D we can describe a rotation in terms of an axis around which the rotation takes place, while in higher than 3-D the notion of a rotation axis is generally not useful: If the rotation takes place in a 2-D plane in p-space, the "axis" is a $(p-2)$-dimensional subspace of fixed points; but if the rotation is more general, the "axis" of fixed points can be of dimensions $p-4$, $p-6$, ... , and such "axes" do not determine a unique 1-parameter family of rotations. A similar point was made by J. W. Tukey (1987, Section 8).

The apparent complexity raised by these issues was answered in a radical way by Asimov's notion of a grand tour (Asimov 1985): Just like 3-D data rotations expose viewers to dynamic 2-D projections of 3-D space, a grand tour exposes viewers to dynamic 2-D projections of higher dimensional space, but unlike 3-D data rotations, a grand tour presents the viewer with an automatic movie of projections with no user control. A grand tour is by definition a movie of low-dimensional projections constructed in such a way that it comes arbitrarily close to any low-dimensional projection; in other words, a grand tour is a space-filling curve in the manifold of low-dimensional projections of high-dimensional data spaces. The grand tour as a fully automatic animation is conceptually simple, but its simplicity leaves users with a mixed experience: that of the power of directly viewing high dimensions, and that of the absence of control and involvement.

Since Asimov's original paper, much of our work on tours has gone into reclaiming the interactive powers of 3-D rotations and extending them in new directions. We did this in several ways: by allowing users to restrict tours to promising subspaces, by offering a battery of view-optimizations, and by re-introducing manual control to the motion of projections. The resulting methods are what we call "guided tours" and "manual tours".

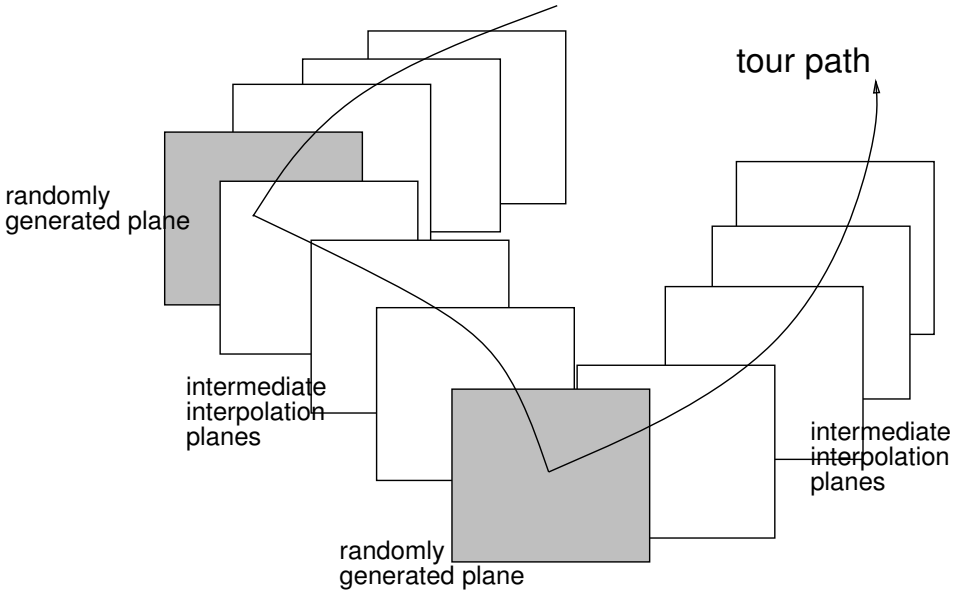

Figure 1: *Schematic depiction of a path of projection planes that interpolates a sequence of randomly generated planes. This scheme is an implementation of a grand tour as used in XGobi (Swayne et al. 1998).*

At the base of these interactively controlled tours is a computational substrate for the construction of paths of projections, and it is this substrate that is the topic of the present article. The simple idea is to base all computations on the continuous interpolation of discrete sequences of projections. An intuitive depiction of this interpolation substrate is shown in Figure 1, and a realistic depiction in Figure 2. Interpolating paths of projections are analogous to connecting line segments that interpolate points in Euclidean spaces. Interpolation provides the bridge between continuous animation and discrete choice of sequences of projections:

- Continuous animation gives viewers a sense of coherence and temporal comparison between pictures seen now and earlier. Animation can be subjected to controls such as start and stop, move forward or back up, accelerate or slow down.

- Discrete sequences offer freedom of use for countless purposes: they can be randomly selected, systematically constrained, informatively optimized, or manually directed. In other words, particular choices of discrete sequences amount to implementations of grand tours, guided tours, and manual tours.

For a full appreciation of the second point, which reflects the power of the proposed interpolation substrate, we list a range of high-level applications it can support. Most of these applications are available in the XGobi software (Swayne et al. 1998) and

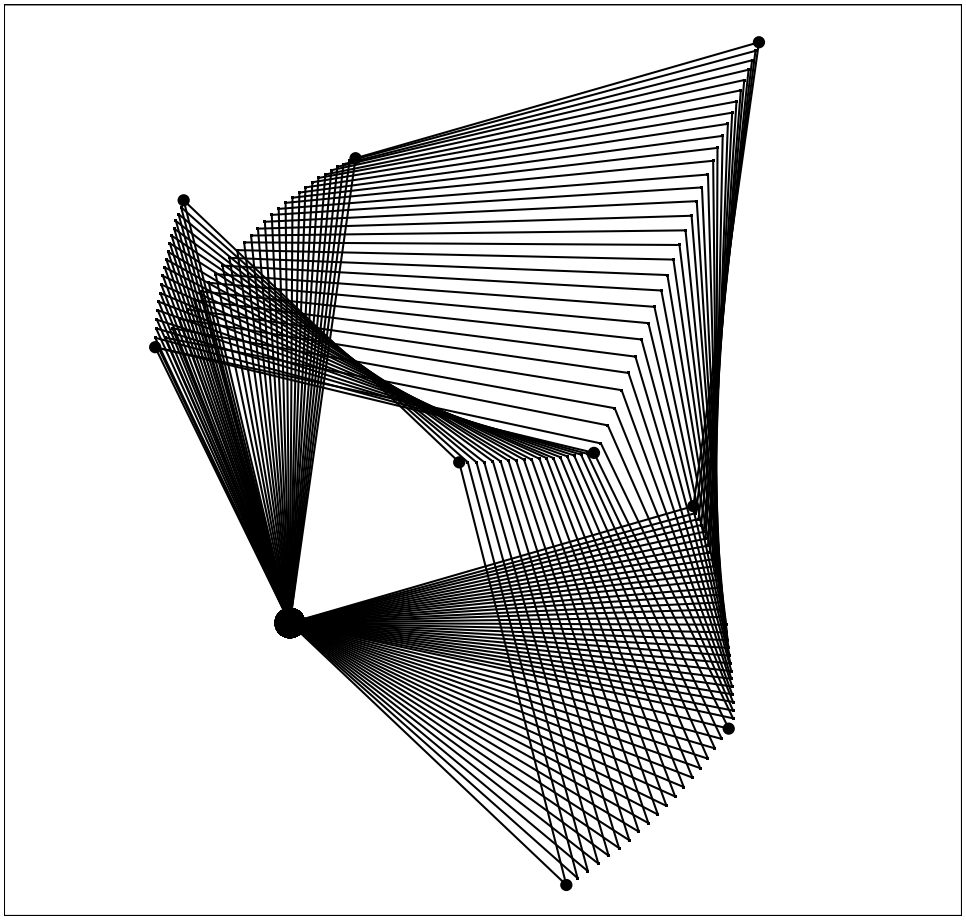

Figure 2: *XGobi looking at XGobi: This figure shows a random projection of a path of projection planes that interpolates three randomly generated projections. The planes are represented as squares spanned by their projection frames. The origin is marked by a fat dot, and the three random planes by small dots. The figure was created in XGobi's tour module with data generated from a sequence of XGobi projection frames.*

its successor, the GGobi software. Sequences of projections can be constructed as follows:

- **Random choice:** If the goal is to "look at the data from all sides," that is, to get an overview of the shape of a p-dimensional pointcloud, it may be useful to pick a random sequence of projections. This choice yields indeed a particular implementation of the **grand tour** (Asimov 1985), defined as an infinite path of projections that becomes dense in the set of all projections. Denseness is trivially achieved by interpolating a discrete set of projections that is dense in the set of projections, the so-called Grassmann manifold.
- **Precomputed choice:** At times one would like to design paths of data projections for specific purposes. An example is the **little tour** (McDonald 1982)

which follows a path that connects all projections onto pairs of variables. The little tour is the animated analog of a scatterplot matrix which also shows plots of all pairs of variables, but without temporal connection. – Another example is what may be called the **packed tour**: One places a fixed number of projections as far as possible from each other — that is, one finds a packing of fixed size of the Grassmann manifold — and places a shortest (Hamiltonian) path through these projections. Such Grassmann packings have been computed by Conway, Hardin and Sloane (1995). The packed tour is essentially an improvement over the random-choice based grand tour under the stipulation that one is only willing to watch the tour for a fixed amount of time (Asimov 1985, remark R1, p. 130).

- **Data-driven choice:** These are methods we summarily call **guided tours** because the paths are partly or completely guided by the data. A particular guided tour method uses *projection pursuit*, a technique for finding data projections that are most structured according to a criterion of interest, such as a clustering criterion or a spread criterion. Our implementation of interactive projection pursuit follows hill-climbing paths on the Grassmann manifold by interpolating gradient steps of various user-selected projection pursuit criteria (Cook et al. 1995). Alternation with the grand tour provides random restarts for the hill-climbing paths. — Another guided tour method uses *multivariate analysis* rather than projection pursuit: One restricts paths of projections for example to principal component and canonical variate subspaces (Hurley and Buja 1990).
- **Manual choice:** Contrary to common belief, it is possible to manually direct projections from higher than three dimensions. Intuitively, this can be done by "pulling variables in and out of a projection." Generally, any direction in p-space can serve as a "lever" for pulling or pushing a projection. The notion is thus to move a projection plane with regard to immediately accessible levers such as variable directions in data space. Unlike customary 3-D controls, this notion applies in arbitrarily many dimensions, resulting in what we may call a **manual tour**. See Cook and Buja (1996) for more details and for an implementation in the XGobi software. A related technique has been proposed by Duffin and Barrett (1994).

While our tendency has been away from random grand tours towards greater human control in guided and manual tours, the emphasis of Wegman (1991) has been to remove the limitation to 1-D and 2-D projections with a powerful proposal of arbitrary k-dimensional tours, where k is the projection dimension. Such projections can be rendered by parallel coordinate plots and scatterplot matrices, but even the full-dimensional case $k = p$ is meaningful, when there is no dimension reduction, only dynamic p-dimensional rotation of the basis in data space.

Another area of extending the idea of tours by Wegman and co-authors (Wegman et al. 1998;; Symanzik et al. 2002) is in the so-called image tour, in which high-dimensional spectral images are subjected to dynamic projections and reduced to 1-

or 2-dimensional images that can be rendered as time-varying gray-scale or false-color images.

A word on the relation of the present work to the original grand tour paper by Asimov (1985) is in order: That paper coined the term "grand tour" and devised the most frequently implemented grand tour algorithm, the so-called "torus method." This algorithm parametrizes projections, constructs a space-filling path in parameter space, and maps this path to the space of projections. The resulting space-filling path of projections has some desirable properties, such as smoothness, reversibility and ease of speed control. There are two reasons, however, why some of us now prefer interpolation methods:

- The paths generated by the torus method can be highly non-uniformly distributed on the manifold of projections. As a result, the tour may linger among projection planes that are near some of the coordinate axes but far from others. Such non-uniformities are unpredictable and depend on subtle design choices in the parametrization. Although the paths are uniformly distributed in parameter space, their images in the manifold of projections are not, and the nature of the non-uniformities is hard to analyze. By contrast, tours based on interpolation of uniformly sampled projections are uniformly distributed by construction.

  Wegman and Solka (2002) make an interesting argument that the problem of non-uniformity is less aggravating for so-called full-dimensional tours (Wegman 1991), in which data space is not projected but subjected to dynamic basis rotations and the result shown in parallel coordinate plots or scatterplot matrices. This argument sounds convincing, and the above criticism of the torus method therefore applies chiefly to tours that reduce the viewing dimension drastically. Still, non-uniformity is a defect even for full-dimensional tours (although more aesthetic than substantive), and there is no reason why one should put up with a defect that can be remedied.

- In interactive visualization systems, for which the grand tour is meant, users have a need to frequently change the set of active variables that are being viewed. When such a change occurs, a primitive implementation of a tour simply aborts the current tour and restarts with another subset of the variables. As a consequence there is discontinuity. In our experience with DataViewer (Buja et al. 1988) and XGobi (Swayne et al. 1998) we found this very undesirable. In order to fully grant the benefits of continuity — in particular visual object and shape persistence — a tour should remain continuous at all times, even when the subspace is being changed. Therefore, when a user changes variables in XGobi, the tour interpolates to the new viewing space in a continuous fashion. The advantages of interpolation methods were thus impressed on us by needs at the level of user perception.

  The above argument applies of course only to tours that keep the projection

dimension fixed, as is the case for X/GGobi's 1-D and 2-D tours. Wegman's (1991) k-dimensional tours, however, permit arbitrary projection dimensions, and when this dimension is changed, continuous transition is more difficult to achieve because projection dimensions are added or removed. However, when the dimension of data space is changed but the projection dimension is preserved, continuous transitions are possible.

This article is layed out as follows: Section 2 gives an algorithmic framework for computer implementations, independent of the particular interpolating algorithms. Section 3 describes an interpolation scheme for projections that is in some sense optimal. Section 4 gives several alternative methods that have other benefits. All methods are based on familiar tools from numerical linear algebra: real and complex eigendecompositions, Givens rotations, and Householder transformations.

Free software that implements dynamic projections can be obtained from the following sites:

- **GGobi** by Swayne, Temple-Lang, Cook, and Buja for Linux and MS Windows™:

  http://www.ggobi.org/

  See Swayne, Buja, and Temple-Lang (2003).

- **XGobi** by Swayne, Cook and Buja (1998) for Unix® and Linux operating systems:

  http://www.research.att.com/areas/stat/xgobi/

  See Swayne, Cook and Buja (1998). The tour module of XGobi implements most of the numerical methods described in this paper. A version of the software that runs under MS Windows™ using a commercial X™ emulator has been kindly provided by Brian Ripley:

  http://www.stats.ox.ac.uk/pub/SWin/

- **CrystalVision** for MS Windows™ by Wegman, Luo and Fu:

  ftp://www.galaxy.gmu.edu/pub/software/

  See Wegman (2003). This software implements among other things k-dimensional (including full-dimensional) tours rendered with parallel coordinates and scatterplot matrices.

- **ExploreN** for SGI Unix® by Luo, Wegman, Carr and Shen:

  ftp://www.galaxy.gmu.edu/pub/software/

  See Carr, Wegman, and Luo (1996).

- **Lisp-Stat** by Tierney contains a grand tour implementation:

  http://lib.stat.cmu.edu/xlispstat/

See Tierney (1990, chapter 10).

Applications: This article is about mathematical and computational aspects of dynamic projections. As such it will leave those readers unsatisfied who would like to see dynamic projection methods in use for actual data analysis. We can satisfy this desire partly by providing pointers to other literature: A few of our own papers give illustrations (Buja, Cook and Swayne, 1996; Cook, Buja, Cabrera and Hurley, 1995; Furnas and Buja, 1994; Hurley and Buja, 1990). A wealth of applications with interesting variations, including image tours, is by Wegman and co-authors. See for example Wegman (1991), Wegman and Carr (1993), Wegman and Shen (1993, Wegman et al. (1998), Symanzik et al. (2002), Wegman and Solka (2002), Wegman (2003).

Three-D data rotations are used extensively in the context of regression by Cook and Weisberg (1994).

It should be kept in mind that the printed paper has never been a satisfactory medium for conveying intuitions about motion graphics. Nothing replaces live or taped demonstrations or, even better, hands-on experience.

Terminology: In what follows we use the terms *"plane"* or *"projection plane"* to denote subspaces of any dimension, not just two.

2 Tools for Constructing Plane and Frame Interpolations: Orthonormal Frames and Planar Rotations

We outline a computational framework that can serve as the base of any data visualization with dynamic projections. We need notation for the linear algebra that underlies projections:

- Let p be the high dimension in which the data live, and let $\boldsymbol{x}_i \in I\!R^p$ denote the column vector representing the i'th data vector ($i = 1, ...N$). The practically useful data dimensions are p from 3 (traditional) up to about 10.
- Let d be the dimension onto which the data is being projected. The typical projection dimension is $d = 2$, as when a standard scatterplot is used to render the projected data. However, the extreme of $d = p$ exists also and has been put to use by Wegman (1991) in his proposal of a full-dimensional grand tour in which the rotated p-space is rendered in a parallel coordinate plot or scatterplot matrix (this is a special case of his general k-dimensional tour; note we use d where he uses k).
- An "orthonormal frame" or simply a "frame" F is a $p \times d$ matrix with pairwise orthogonal columns of unit length:

  $$F^T F = I_d \ ,$$

  where I_d is the identity matrix in d dimensions, and F^T is the transpose of F. The orthonormal columns of F are denoted by $\boldsymbol{f}_i$ ($i = 1, ..., d$).

- The projection of $\boldsymbol{x}_i$ onto F is the d-vector $\boldsymbol{y}_i = F^T \boldsymbol{x}_i$. This is the appropriate notion of projection for computer graphics where the components of $\boldsymbol{y}_i$ are used as coordinates of a rendering as scatterplot or parallel coordinate plot on a computer screen.

  [By comparison, d-dimensional projections in the mathematical sense are linear idempotent symmetric rank-d maps; they are obtained from frames F as $P = FF^T$. In this sense the projection of $\boldsymbol{x}_i$ is $P\boldsymbol{x}_i = F\boldsymbol{y}_i$, which is a p-vector in the column space of F. Two frames produce the same mathematical projection iff their column spaces are the same.]

- Paths of projections are given by continuous *one-parameter families* $F(t)$ where $t \in [a, z]$, some real interval representing essentially time. We denote the starting and the target frame by $F_a = F(a)$ and $F_z = F(z)$, respectively.

- The animation of the projected data is given by a path $\boldsymbol{y}_i(t) = F(t)^T \boldsymbol{x}_i$ for each data point $\boldsymbol{x}_i$. At time t, the viewer of the animation sees a rendition of the d-dimensional points $\boldsymbol{y}_i(t)$, such as a scatterplot if $d = 2$ as in XGobi's 2-D tour, or a parallel coordinate plot if $d = p$ as in Wegman's full-dimensional tour, a special case of his general k-dimensional tour (we use d where he uses k).

- We need notation for the dimensions of various subspaces that are relevant for the construction of paths of projections. We write span(...) for the vector space spanned by the columns contained in the arguments. Letting

  $$d_S = \dim(\text{span}(F_a, F_z)) \quad \text{and} \quad d_I = \dim(\text{span}(F_a) \cap \text{span}(F_z)) \ ,$$

  it holds

  $$d_S = 2d - d_I \ .$$

  Some special cases:

  - We have $d_S = 2d$ whenever $d_I = 0$, that is, the starting and target plane intersect only at the origin, which is the generic case for a 2-D tour in $p \geq 4$ dimensions.
  - Tours in $p = 3$ dimensions are just 3-D data rotations, in which case generally $d_S = 3$ and $d_I = 1$.
  - In the other extreme, when the two planes are identical and the plane is rotated within itself, we have $d_S = d_I = d$, which is the generic case for a full-dimensional tour with $p = d_S = d_I$.

2.1 Minimal Subspace Restriction

The point of the following considerations is to reduce the problem of path construction to the smallest possible subspace and allow for particular bases in which interpolation can be carried out simply.

A path of frames $F(t)$ that interpolates two frames F_a and F_z should be parsimonious in the sense that it should not traverse parts of data space that are unrelated to

F_a and F_z. For example, if both these frames exist in the space of variables $x_1, \ldots, x_5$, then the path $F(t)$ should not make use of variable x_6. In general terms, $F(t)$ should live in the joint span $\mathrm{span}(F_a, F_z)$. The restriction of the frame path $F(t)$ to this subspace requires some minor infrastructure which is set up in a step that we call "preprojection".

Preprojection is carried out as follows: Form an arbitrary orthonormal basis of $\mathrm{span}(F_a, F_z)$, for example, by applying Gram-Schmidt to F_z with regard to F_a. Denote the resulting basis frame of size $p \times d_S$ by

$$B = (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{d_S})$$

Note that when $\mathrm{span}(F_a) = \mathrm{span}(F_z) = \mathrm{span}(F_a, F_z)$ and hence $d_S = d_I = d$, the plane is rotated within itself, yet preprojection is *not* vacuous: Most interpolation algorithms require particular (for example canonical) bases in order to simplify interpolation.

We can now express the original frames in this basis:

$$F_a \;=\; BW_a \quad \text{and} \quad F_z \;=\; BW_z \;,$$

where $W_a = B^T F_a$ and $W_z = B^T F_z$ are orthonormal frames of size $d_S \times d$. The problem is now reduced to the construction of paths of frames $W(t)$ that interpolate the preprojected frames W_a and W_z. The corresponding path in data space is

$$F(t) \;=\; BW(t)$$

and the viewing coordinates of a data vector $\boldsymbol{x}_i$ are

$$\boldsymbol{y}_i(t) \;=\; F(t)^T \boldsymbol{x}_i \;=\; W^T(t) B^T \boldsymbol{x}_i \;.$$

If one anticipates projections from very high dimensions, it might be useful to preproject the data to $\boldsymbol{\xi}_i = B^T \boldsymbol{x}_i$ and lessen the computational expense by computing only $\boldsymbol{y}_i(t) = W^T(t)\boldsymbol{\xi}_i$ during animation. When the data dimension p is below 10, say, the reduction in computational cost may not be worth the additional complexity.

2.2 Planar rotations

The basic building blocks for constructing paths of frames are planar rotations, that is, rotations that have an invariant 2-D plane with action corresponding to

$$\begin{pmatrix} c_\tau & -s_\tau \\ s_\tau & c_\tau \end{pmatrix}$$

and an orthogonal complement of fixed points. ($c_\tau = \cos(\tau)$ and $s_\tau = \sin(\tau)$).

If the action is in the plane of variables i and j, we denote the rotation by $R_{ij}(\tau)$, which is then also called a *Givens rotation*. Note that the order of i and j matters:

$R_{ij}(\tau) = R_{ji}(-\tau)$. For efficiency, $R_{ij}(\tau)$ is never stored explicitly; matrix multiplications involving $R_{ij}(\tau)$ are directly computed from i, j, and τ. See Golub and Van Loan (1983), Section 3.4, for computational details (note that their $J(i,k,\theta)$ is our $R_{ik}(-\theta)$).

The methods we introduce in the following sections are based on the composition of a number of Givens rotations in a suitable coordinate system. The basic step is always the construction of a composition that maps the starting frame onto the target frame. Writing $R_\mu(\tau_\mu)$ for $R_{i_\mu j_\mu}(\tau_\mu)$, this is

$$W_z \;=\; R_m(\tau_m) \;\ldots\; R_2(\tau_2)\; R_1(\tau_1)\; W_a$$

in the preprojection. We arrive at an interpolating path of frames by simply inserting a time parameter t into the formula:

$$W(t) \;=\; R_m(\tau_1 t) \;\ldots\; R_2(\tau_2 t)\; R_1(\tau_1 t)\; W_a \;,$$

where $0 \le t \le 1$. Obviously, $W(0) = W_a$ and $W(1) = W_z$. In compact notation we write

$$R(\boldsymbol{\tau}) \;=\; R_m(\tau_m) \;\ldots\; R_2(\tau_2)\; R_1(\tau_1) \;, \qquad \boldsymbol{\tau} = (\tau_1, \ldots, \tau_m) \;.$$

Interpolating paths based on rotations are never unique. For one thing, if $W_z = R(\boldsymbol{\tau})\, W_a$, then any other $\tilde{\boldsymbol{\tau}}$ with $\tilde{\tau}_j = \tau_j + k_j \cdot 2\pi$ (where k_j are integers) also satisfies $W_z = R(\tilde{\boldsymbol{\tau}})\, W_a$. Among all these candidates, one usually selects the $\boldsymbol{\tau}$ that is closest to the vector of zero angles.

Now the raw parameter $t \in [0,1]$ is of course not a good choice for creating an animation: if one were to move in equi-distant steps $t_i = \Delta \cdot i$ from 0 to 1, one would move at differing speeds, depending on how far the starting and target frames are apart. The speed would be slow for nearby frames and fast for distant ones. In addition, there are some subtle issues of non-constancy of speed when the sequence of Givens rotations is complex.

Because speed is an issue that is general and independent of the choice of path of frames, we deal with it before we describe specific choices of paths.

2.3 Calculation and control of speed

Assuming that a speed measure for moving frames has been chosen, we can step along paths of frames in such a way that the motion has constant speed with regard to the chosen measure, thereby providing constancy of motion. What measure of speed should be chosen? We cannot discuss here the full answer, but an outline is as follows.

Speed is distance traveled per unit of time. For paths of frames $F(t)$ the speed at time t is therefore some function of the derivative $F'(t)$. The question is what the natural speed measures are. A natural requirement is certainly invariance under orthogonal coordinate transformations a rotated path should have the same speed

characteristics as the unrotated path. Surprisingly this requirement alone is powerful enough to constrain the speed measures to the following family:

$$g_F(F') \;=\; \alpha_p \cdot \|F'\|^2_{Frob} \;+\; (\alpha_w - \alpha_p) \cdot \|F^T F'\|^2_{Frob} \;. \tag{1}$$

where $\alpha_p > 0$ and $\alpha_w \geq 0$, and $\|A\|^2_{Frob} = \sum_{i,j} A^2_{i,j} = \text{trace}(A^T A) = \text{trace}(AA^T)$ is the Frobenius norm of matrices. In terms of differential geometry, these are all possible rotation-invariant Riemannian metrics on the so-called Stiefel manifold of frames. See Theorem 2 of Buja et al. (2004). We cannot go into the details, but here is some intuition: FF^T being the orthogonal projection onto span(F) and $I - FF^T$ onto its orthogonal complement, the quantity $\|(FF^T)F'\|^2_{Frob} = \|F^T F'\|^2_{Frob}$ measures how fast the path rotates **within** the projection plane, whereas $\|(I - FF^T)F'\|^2_{Frob} = \|F'\|^2_{Frob} - \|F^T F'\|^2_{Frob}$ measures how fast the path rotates **out of** the current projection plane. In this light, Equation (1) takes the more interpretable form

$$g_F(F') \;=\; \alpha_p \cdot \|(I - FF^T)\, F'\|^2_{Frob} \;+\; \alpha_w \cdot \|(FF^T)\, F'\|^2_{Frob} \;. \tag{2}$$

According to this formula any squared speed measure can be obtained as a positive linear combination of squared out-of-plane and within-plane speeds.

Even with the reduction to the above family of speed measures, we are still left with many choices. Measures that differ only by a constant positive factor are equivalent, however, hence the remaining choice is that of the ratio α_w/α_p. In Buja et al. (2004) we single out two choices for which arguments can be given: $\alpha_p = \alpha_w = 1$, which is the simplest in algebraic terms, and $\alpha_p = 2$, $\alpha_w = 1$, which is mathematically most meaningful. The latter gives a greater weight to out-of-plane speed than the former. Either choice works reasonably well in practice.

In order to control speed in computer implementations, the following considerations are elementary: Computer animations are generated by updating the display at a fast but *constant* rate (at least 5-10 per second). This implies that animation speed is not usually controlled by varying the update rate but by varying the step size along the motion path: Wider steps produce faster motion.

For dynamic projections, this implies that a path of frames is discretized, and speed is controlled by proper choice of the step size of the discretization. Incrementally, this means that at time t, one steps from $F(t)$ to $F(t + \Delta)$, amounting to a distance $\int_t^{t+\Delta} g_F(F'(\tau))^{1/2} d\tau$. Approximate constancy of speed can be provided as follows. Using the first order approximation

$$\text{StepSize} \;=\; \int_t^{t+\Delta} g_{F(\tau)}(F'(\tau))^{1/2} d\tau \;\approx\; g_{F(t)}(F'(t))^{1/2} \cdot \Delta \;,$$

we can choose the increment Δ as

$$\Delta \;=\; \text{StepSize}/g_{F(t)}(F'(t))^{1/2} \;,$$

where StepSize is a user-chosen speed parameter that expresses something akin to "degrees of motion between updates." Typically, the user does not need to know actual values of StepSize because this quantity is controlled interactively through graphical gauges. [As a recommendation to implementors, such gauges such as sliders should not represent StepSize linearly: At slow speeds it is important to be offered very precise control, while for high speeds it is more important to be offered a large range than high precision. Letting StepSize be proportional to the square of the value read from a slider works well.]

We note that all speed calculations can be carried out in the preprojection: Because $B^T B = I_{d_S}$ we have $g_F(F'(t)) = g_W(W'(t))$ for the invariant speed measures of Equation (1).

Abbreviating the concatenated planar rotations of Section 2.2 and their derivatives by $R = R(\boldsymbol{\tau} t)$ and $R' = d/dt\ R(\boldsymbol{\tau} t)$, respectively, we have $W' = R' W_a$, and the speed measures of Equation (1) in the form given above become

$$g_W(W') \ = \ \alpha_p \cdot \|R' W_a\|^2_{Frob} \ + \ (\alpha_w - \alpha_p) \cdot \|W_a^T R^T R' W_a\|^2_{Frob} \ .$$

When R is a complex concatenation of many planar rotations, it can be impossible or at least messy to obtain R' analytically. It may then be advantageous to approximate R' numerically by actually computing the matrix $R(\boldsymbol{\tau} t)$ for two close values of t, and calculating a finite difference quotient,

$$R' \ \approx \ \left(R(\boldsymbol{\tau}(t+\delta)) - R(\boldsymbol{\tau} t)\right)/\delta$$

which can be substituted for R'. In some cases, such as the optimal paths of Sections 3 and 4.1, there exist explicit formulas for speed.

2.4 Outline of an algorithm for interpolation

In this section we give the schematic outline of an interpolation algorithm. Given starting and target frames F_a and F_z, the outline assumes that we know how to construct a preprojection basis B and a sequence of planar rotations $R(\boldsymbol{\tau}) = R_m(\tau_m) \ldots R_1(\tau_1)$ that map $W_a = B^T F_a$ to $W_z = B^T F_z$. The construction of B and $R(\boldsymbol{\tau})$ is dependent on the interpolation method, and in the following sections we give several such constructions.

Algorithm:

1. *Given a starting frame F_a, create a new target frame F_z.*
2. *Initialize interpolation:*
   - *Construct a preprojection basis B of* $\mathrm{span}(F_a, F_z)$*, where B has d_S columns.*
   - *Check: If starting and target plane are the same: $d = d_S$, and if starting and target frame have opposite orientations:* $\det(F_a^T F_z) = -1$*, then interpolation within the span is impossible.*
     Possible remedy: flip one frame vector, $\boldsymbol{f}_{z,d} \leftarrow -\boldsymbol{f}_{z,d}$.

- *Preproject the frames:* $W_a = B^T F_a$, $W_z = B^T F_z$.
- *Construct planar rotations in coordinate planes:*
  $R(\boldsymbol{\tau}) = R_m(\tau_m) \dots R_1(\tau_1)$, $\boldsymbol{\tau} = (\tau_1, \dots, \tau_m)$,
  such that $W_z = R(\boldsymbol{\tau}) W_a$.
- *Initialize:* $t \leftarrow 0$.

3. *Execute interpolation; iterate the following:*

   $t \leftarrow \min(1, t)$
   $W(t) \leftarrow R(\boldsymbol{\tau} t) W_a$
   $F(t) \leftarrow BW(t)$ $(render\ frame)$
   $\boldsymbol{y}_i(t) \leftarrow F(t)^T \boldsymbol{x}_i$, $i = 1, ..., N$ $(render\ data)$
   $If\ \ t = 1:$ *break iterations.*
   $Else:\ \ \Delta \leftarrow \text{StepSize}/g_W(W')^{1/2}$, $t \leftarrow t + \Delta$, *do next iteration.*

4. *Set* $F_a \leftarrow F_z$ *and go to beginning.*

In the following sections, we will only specify the construction of B and $R(\boldsymbol{\tau})$ and refer the reader to the above algorithm.

When the algorithm is integrated in a larger interactive visualization system such as XGobi or GGobi, a number of additional measures have to be taken at each iteration because interactions by the user may have reset some of the parameters:

- Read the speed parameter StepSize.
- Check whether a new subspace of data space has been selected. Most often a new subspace is specified in terms of a new subset of data variables.
- Check whether another method of target generation has been selected.

If one of the last two checks is positive, the current path has to be interrupted and a new path has to be initialized with the proper changes.

The two lines marked "*render*" imply execution of the rendering mechanisms for generating a new graphical scene from the data projection, such as drawing a new scatterplot or parallel coordinate plot, and for giving the viewer feedback about the position of the current projection, such as drawing a p-pod or generalized tripod that represents the projection of the canonical variable basis onto the current plane.

In what follows we describe the details for algorithms that interpolate frames F_a and F_z (Section 4), but prior we describe an algorithm that interpolates planes (Section 3) in the sense that only span(F_z) is specified and the particular frame F_z within the target plane is determined by the algorithm. The paths produced by the algorithm have the desirable property that their within-plane rotation is zero: $F^T F' = 0$.

3 Interpolating Paths of Planes

We describe paths of frames that optimally interpolate planes in the sense that they are locally shortest (geodesic) with regard to the metrics discussed in Section 2.3.

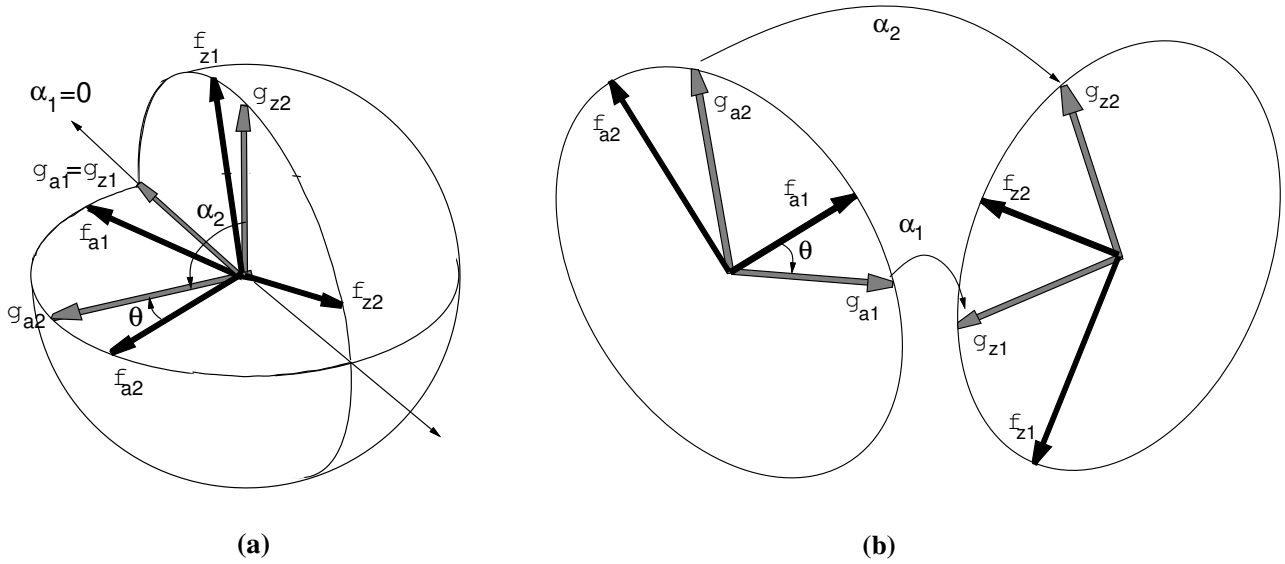

Figure 3: *Relative positions of given orthonormal bases,* $(\boldsymbol{f}_{a,1}, \boldsymbol{f}_{a,2})$, $(\boldsymbol{f}_{z,1}, \boldsymbol{f}_{z,2})$, *and pairs of principal directions,* $(\boldsymbol{g}_{a,1}, \boldsymbol{g}_{a,2})$, $(\boldsymbol{g}_{z,1}, \boldsymbol{g}_{z,2})$, *of the two 2-planes in (a)* 3 *dimensions, (b)* 4 *dimensions (the origin is pulled apart to disentangle the picture).*

The interpolation scheme based on these paths is appropriate for rendering methods that are visually invariant under changes of orientation (Buja et al. 2004). The construction of the paths is based on the following:

Fact: *Given two planes of dimension* d*, there exist orthonormal* d*-frames* G_a *and* G_z *that span the planes and for which* $G_a^T G_z$ *is diagonal.*

Optimal paths of planes essentially rotate the columns of G_a into the corresponding columns of G_z. Denoting the columns of G_a and G_z by $\boldsymbol{g}_{a,j}$ and $\boldsymbol{g}_{z,j}$, respectively, we see that the planes spanned by $\boldsymbol{g}_{a,j}$ and $\boldsymbol{g}_{z,j}$ are mutually orthogonal for $j = 1, 2, ..., d$, and they are either 1-dimensional (if $\boldsymbol{g}_{a,j} = \boldsymbol{g}_{z,j}$) or 2-dimensional (otherwise). The motion is carried out by a moving frame $G(t) = (\boldsymbol{g}_1(t), ..., \boldsymbol{g}_d(t))$ as follows: If $\boldsymbol{g}_{a,j} = \boldsymbol{g}_{z,j}$, then $\boldsymbol{g}_j(t) = \boldsymbol{g}_{a,j}$ is at rest; otherwise $\boldsymbol{g}_j(t)$ rotates from $\boldsymbol{g}_{a,j}$ to $\boldsymbol{g}_{z,j}$ at constant speed proportional to the angle between $\boldsymbol{g}_{a,j}$ and $\boldsymbol{g}_{z,j}$.

The columns of the frames G_a and G_z are called "principal directions" of the pair of planes. Without loss of generality, we can assume that the diagonal elements of $G_a^T G_z$ are 1) non-negative and 2) sorted in descending order: $1 \geq \lambda_j \geq \lambda_{j+1} \geq 0$. (For non-negativity, multiply a column of G_a with -1 if necessary.) The diagonal elements λ_j of $G_a^T G_z$, called "principal cosines", are the stationary values of the cosines of angles between directions in the two planes. In particular, the largest principal cosine describes the smallest angle that can be formed with directions in the two planes. The angles $\alpha_j = \cos^{-1} \lambda_j$ are called principal angles ($0 \leq \alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_d \leq \pi/2$). Note that for two planes with a nontrivial intersection of dimension $d_I > 0$, there will

be d_I vanishing principal angles: $\alpha_1 = \ldots = \alpha_{d_I} = 0$. In particular, two 2-planes in 3-space always have at least an intersection of dimension $d_I = 1$, and hence $\lambda_1 = 1$ and $\alpha_1 = 0$. The geometry of principal directions is depicted in Figure 3.

[The principal angles are the essential invariants for the relative position of two planes to each other. This means technically that two pairs of planes with equal principal angles can be mapped onto each other with a rotation (Halmos 1970).]

Principal directions are easily computed with a singular value decomposition (SVD). Given two arbitrary frames F_a and F_z spanning the respective planes, let

$$F_a^T F_z \;=\; V_a \Lambda V_z^T$$

be the SVD of $F_a^T F_z$, where V_a and V_z are $d \times d$ orthogonal matrices and Λ is diagonal with singular values λ_j in the diagonal. The frames

$$G_a \;=\; F_a V_a \quad \text{and} \quad G_z \;=\; F_z V_z$$

satisfy $G_a^T G_z = \Lambda$. Hence the singular values are just the principal cosines, and the orthogonal transformations V_a and V_z provide the necessary rotations of the initial frames. (See Bjorck and Golub (1973), and Golub and Van Loan (1983) Section 12.4.).

The following construction selects an interpolating path with zero within-plane spin that is not only locally but globally shortest. The path is unique iff there are no principal angles of 90 degrees.

Path Construction:

1. Given a starting frame F_a and a preliminary frame F_z spanning the target plane, compute the SVD

   $$F_a^T F_z \;=\; V_a \Lambda V_z^T \;, \quad \Lambda = \mathrm{diag}(\lambda_1 \geq \ldots \geq \lambda_d) \;,$$

   and the frames of principal directions:

   $$G_a \;=\; F_a V_a \;, \quad G_z \;=\; F_z V_z \;.$$

2. Form an orthogonal coordinate transformation U by, roughly speaking, orthonormalizing (G_a, G_z) with Gram-Schmidt. Due to $G_a^T G_z = \Lambda$, it is sufficient to orthonormalize $\boldsymbol{g}_{z,j}$ with regard to $\boldsymbol{g}_{a,j}$, yielding $\boldsymbol{g}_{*,j}$. This can be done only for $\lambda_j < 1$ because for $\lambda_j = 1$ we have $\boldsymbol{g}_{a,j} = \boldsymbol{g}_{z,j}$, spanning the d_I-dimensional intersection $\mathrm{span}(F_a) \cap \mathrm{span}(F_z)$.
   (Numerically, use the criterion $\lambda_j > 1 - \epsilon$ for some small ϵ to decide inclusion in the intersection.)
   Form the preprojection basis

   $$B \;=\; (\boldsymbol{g}_{a,d}, \boldsymbol{g}_{*,d},\; \boldsymbol{g}_{a,d-1}, \boldsymbol{g}_{*,d-1},\; \ldots, \boldsymbol{g}_{a,d_I+1}, \boldsymbol{g}_{*,d_I+1},\; \boldsymbol{g}_{a,d_I}, \boldsymbol{g}_{a,d_I-1}, \ldots, \boldsymbol{g}_{a,1}) \;.$$

   The last d_I vectors $\boldsymbol{g}_{a,d_I}, \boldsymbol{g}_{a,d_I-1}, \ldots, \boldsymbol{g}_{a,1}$ together span the intersection of the starting and target plane. The first $d - d_I$ pairs $\boldsymbol{g}_{a,j},\; \boldsymbol{g}_{*,j}$ span each a 2-D plane in which we perform a rotation:

3. The sequence of planar rotations $R(\boldsymbol{\tau})$ is composed of the following $d-d_I$ planar rotations:

$$R_{12}(\tau_1)R_{34}(\tau_2)\dots\ , \quad \text{where} \quad \tau_j = \cos^{-1}\lambda_{d+1-j} \quad \text{for} \quad j = 1, 2, \dots, d-d_I.$$

The resulting path moves G_a to G_z, and hence $F_a = G_a V_a^T$ to $G_z V_a^T$. The original frame $F_z = G_z V_z^T$ in the target plane is thus replaced with the target frame $G_z V_a^T = F_z V_z V_a^T$. The rotation $V_z V_a^T$ maps F_z to the actual target frame in span(F_z).

For these paths, it is possible to give an explicit formula for speed measures, of which there exists essentially only one: In the preprojection basis, the moving frame is of the form

$$W(t) \ = \ R(\boldsymbol{\tau}t)W_a \ = \ \begin{pmatrix} c_{\tau_1 t} & 0 & \dots \\ s_{\tau_1 t} & 0 & \dots \\ 0 & c_{\tau_2 t} & \dots \\ 0 & s_{\tau_2 t} & \dots \\ \dots & \dots & \dots \end{pmatrix} ,$$

hence $g_W(W') = \|W'\|^2 = \tau_1^2 + \tau_2^2 + \dots$. In the formula for speed measures, Equation (2), the second term for within-plane rotation vanishes due to $F^T F' = 0$. The spead measure is therefore the same for all choices of α_w. The speed measure $g_W(W')$ is constant along the path and therefore needs to be computed only once at the beginning of the path.

In the most important case of $d = 2$-dimensional projections, the SVD problem is of size 2×2, which can be solved explicitly: The eigenvalues of the 2×2 symmetric matrix $(F_a^T F_z)(F_a^T F_z)^T$ are the squares of the singular values of $F_a^T F_z$ and can be found by solving a quadratic equation. We give the results without proof:

Lemma 1:
The principal directions in a 2-D starting plane are given by

$$\boldsymbol{g}_{a,1} = \cos\theta \cdot \boldsymbol{f}_{a,1} + \sin\theta \cdot \boldsymbol{f}_{a,2} \ , \quad \boldsymbol{g}_{a,2} = -\sin\theta \cdot \boldsymbol{f}_{a,1} + \cos\theta \cdot \boldsymbol{f}_{a,2} \ ,$$

where

$$\tan(2\theta) = 2\frac{(\boldsymbol{f}_{a,1}^T \boldsymbol{f}_{z,1}) \cdot (\boldsymbol{f}_{a,2}^T \boldsymbol{f}_{z,1}) + (\boldsymbol{f}_{a,1}^T \boldsymbol{f}_{z,2}) \cdot (\boldsymbol{f}_{a,2}^T \boldsymbol{f}_{z,2})}{(\boldsymbol{f}_{a,1}^T \boldsymbol{f}_{z,1})^2 + (\boldsymbol{f}_{a,1}^T \boldsymbol{f}_{z,2})^2 - (\boldsymbol{f}_{a,2}^T \boldsymbol{f}_{z,1})^2 - (\boldsymbol{f}_{a,2}^T \boldsymbol{f}_{z,2})^2} \ . \tag{3}$$

The principal cosine λ_j is obtained by projecting $\boldsymbol{g}_{a,j}$ onto the target plane: $\lambda_j^2 = (\boldsymbol{g}_{a,j}^T \boldsymbol{f}_{z,1})^2 + (\boldsymbol{g}_{a,j}^T \boldsymbol{f}_{z,2})^2$.

Caution is needed when using Equation (3): It has four solutions in the interval $0 \le \theta < 2\pi$ spaced by $\pi/2$. Two solutions spaced by π yield the same principal direction up to a sign change. Hence the four solutions correspond essentially to two principal directions.

The denominator of the right hand side of Equation (3) is zero when the principal angles of the two planes are identical, in which case all unit vectors in the two planes are principal.

The principal 2-frame in the target plane should always be computed by projecting the principal 2-frame of the starting plane. If both principal angles are $\pi/2$, any 2-frame in the target plane can be used for interpolation. If only one of the principal angles is $\pi/2$, one obtains $\boldsymbol{g}_{z,2}$ as an orthogonal complement of $\boldsymbol{g}_{z,1}$ in the target plane.

4 Interpolating Paths of Frames

Frame interpolation — as opposed to plane interpolation — is necessary when the orientation of the projection matters, as in full-dimensional tours. In addition, frame interpolation can always be used for plane interpolation when the human cost of implementing paths with zero within-plane spin is too high. Some frame interpolation schemes are indeed quite simple to implement. They have noticeable within-plane spin — a fact which XGobi users can confirm by playing with interpolation options in the tour module.

The methods described here are based on 1) decompositions of orthogonal matrices, 2) Givens decompositions, and 3) Householder decompositions. The second and third of these methods do not have any optimality properties, but the first method is optimal for full-dimensional tours in the sense that it yields geodesic paths in $SO(p)$.

4.1 Orthogonal matrix paths and optimal paths for full-dimensional tours

The idea of this interpolation technique is to augment the starting frame and the target frame to square orthogonal matrices and solve the interpolation problem in the orthogonal group ($SO(d_S)$, to be precise). The implementation is quite straightforward. The suboptimality of the paths is obvious from the arbitrariness of the way the frames are augmented to square orthogonal matrices. This is why full-dimensional paths are optimal: They do not require any augmentation at all. Strictly speaking, the requirement for optimality is not "full dimensionality" in the sense $d = p$, but simply that the starting plane and the target plane are the same, that is, $d_S = d$, in which case a simple rotation of the resting space takes place. For lower-dimensional frames whose space is not at rest ($d < d_S$), this arbitrariness can be remedied at least for the metric defined by $\alpha_w = 1$ and $\alpha_p = 2$: The trick is to optimize the augmentation (Asimov and Buja 1994). — The method is based on the following:

Fact: *For any orthogonal transformation A with* $\det(A) = +1$*, there exists an orthogonal matrix V such that*

$$A \;=\; VR(\boldsymbol{\tau})V^T \quad \textit{where} \quad R(\boldsymbol{\tau}) \;=\; R_{12}(\tau_1)\; R_{34}(\tau_2)\; \ldots\; .$$

That is, in a suitable coordinate system, every orthogonal transformation with det =

$+1$ is the composition of planar rotations in mutually orthogonal 2-D planes:

$$A(\boldsymbol{v}_1, \boldsymbol{v}_2) = (\boldsymbol{v}_1, \boldsymbol{v}_2) \begin{pmatrix} c_\phi & -s_\phi \\ s_\phi & c_\phi \end{pmatrix} ,$$

where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ form an orthonormal basis of an invariant plane. See, for example, Halmos (1958, Section 81).

Invariant planes can be found with a complex eigendecomposition of A (as implemented, for example, in subroutine "dgeev" in LAPACK, available from *netlib.lucent.com/netlib/lapack*). If $\boldsymbol{v} = \boldsymbol{v}_r + i\boldsymbol{v}_i$ is a complex eigenvector of A with eigenvalue $e^{i\phi}$, then the complex conjugate $\bar{\boldsymbol{v}} = \boldsymbol{v}_r - i\boldsymbol{v}_i$ is an eigenvector with eigenvalue $e^{-i\phi}$, hence

$$A(\boldsymbol{v}_r, \boldsymbol{v}_i) \;=\; (\boldsymbol{v}_r, \boldsymbol{v}_i) \begin{pmatrix} c_\phi & s_\phi \\ -s_\phi & c_\phi \end{pmatrix} ,$$

which implies that $-\phi$ is the rotation angle in the invariant plane spanned by the frame $(\boldsymbol{v}_r, \boldsymbol{v}_i)$. (The vectors $\boldsymbol{v}_r$ and $\boldsymbol{v}_i$ need to be normalized as they are not usually of unit length when returned by a routine such as "dgeev".)

Path Construction:

1. Form a preliminary basis frame $\tilde{B}$ for preprojection by orthonormalizing the combined frame (F_a, F_z) with Gram-Schmidt. The preprojection of F_a is

   $$\tilde{W}_a \;=\; \tilde{B}^T F_a \;=\; E_d = ((1,0,0,\ldots)^T, (0,1,0,\ldots)^T, \ldots) .$$

   The target frame $\tilde{W}_z = \tilde{B}^T F_z$ is of a general form.

2. Expand $\tilde{W}_z$ to a full orthogonal matrix A with $\det(A) = +1$, for example, by appending random vectors to $\tilde{W}_z$ and orthonormalizing them with Gram-Schmidt. Flip the last vector to its negative if necessary to ensure $\det(A) = +1$. Note that

   $$\tilde{W}_z \;=\; A\tilde{W}_a ,$$

   because $\tilde{W}_a$ extracts the first d columns from A, which is just $\tilde{W}_z$.

3. Find the canonical decomposition $A = VR(\boldsymbol{\tau})V^T$ according to the above.

4. Change the preprojection basis: $B = \tilde{B}V$, such that $W_a = B^T F_a = V^T \tilde{W}_a$ and $W_z = B^T F_z = V^T \tilde{W}_z$. From the canonical decomposition follows

   $$W_z \;=\; R(\boldsymbol{\tau}) W_a .$$

The above is the only method described in this paper that has not been implemented and tested in XGobi.

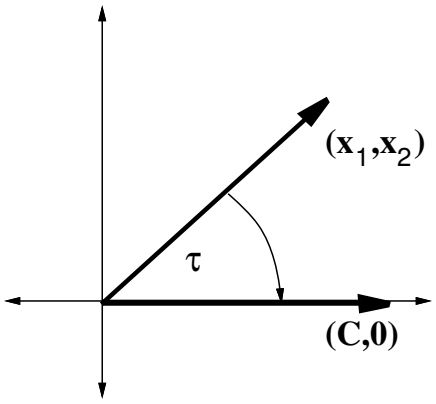

Figure 4: *Rotation to make subvector of matrix coincide with one of the coordinate axes.*

4.2 Givens paths

This interpolation method adapts the standard matrix decomposition techniques based on Givens rotations. While we use Givens rotations for interpolation, Asimov's original grand tour algorithm (Asimov 1985) uses them to (over-)parametrize the manifold of frames. Wegman and Solka (2002 call methods based on Givens rotations "winding algorithms". At the base of all uses of Givens rotations is the following:

Fact: *In any vector* $\boldsymbol{u}$ *one can zero out the i'th coordinate with a Givens rotation in the* (i, j)*-plane for any* $j \neq i$*. This rotation affects only coordinates* i *and* j *and leaves all coordinates* $k \neq i, j$ *unchanged.*

For example, to zero out coordinate x_2 in the (x_1, x_2)-plane, use a rotation $\begin{pmatrix} c_\tau & -s_\tau \\ s_\tau & c_\tau \end{pmatrix}$ with $c_\tau = x_1/(x_1^2 + x_2^2)^{1/2}$ and $s_\tau = -x_2/(x_1^2 + x_2^2)^{1/2}$. That is, τ is the angle from (x_1, x_2) to $(1, 0)$. See Figure 4 for a depiction. For computational details see Golub and Van Loan (1983), Section 3.4.

Sequences of Givens rotations can be used to map any orthonormal d-frame F in p-space to the standard d-frame $E_d = ((1, 0, 0, \ldots)^T, (0, 1, 0, \ldots)^T, \ldots)$ as follows:

- Apply a sequence of $p - 1$ Givens rotations to zero out coordinates $2, \ldots, p$ of the first vector $\boldsymbol{f}_1$. Examples of suitable sequences are rotations in the variables $(1, 2),\ (1, 3),\ (1, 4), \ldots,\ (1, p)$, or $(p - 1, p), \ldots,\ (3, 4),\ (2, 3),\ (1, 2)$, where the second variable is the one whose coordinate is being zeroed out. Care should be taken that the last rotation chooses among the suitable angles τ and $\tau + \pi$

the one that results in the first coordinate =+1. The resulting frame will have $\boldsymbol{e}_1 = (1, 0, 0, \ldots)^T$ as its first vector.

- Apply to the resulting frame a sequence of $p - 2$ Givens rotations to zero out the coordinates $3, \ldots, p$. Do not use coordinate 1 in the process, to ensure that the first vector $\boldsymbol{e}_1$ remains unchanged. A suitable sequence of planes could be $(2, 3),\ (2, 4), \ldots,\ (2, p)$, or else $(p - 1, p), \ldots,\ (3, 4),\ (2, 3)$. Note that the zeros of the first column remain unaffected because $(0, 0)$ is a fixed point under all rotations.
- And so on, till a sequence of $(p-1)+(p-2)+\ldots+(p-d) = pd - \begin{pmatrix} d \\ 2 \end{pmatrix}$ Givens rotations is built up whose composition maps F to E_d.

Path Construction:

1. Construct a preprojection basis B by orthonormalizing F_z with regard to F_a with Gram-Schmidt:
$$B \ = \ (F_a, F_*) \ .$$
2. For the preprojected frames
$$W_a \ = \ B^T F_a \ = \ E_d \quad \text{and} \quad W_z \ = \ B^T F_z$$
construct a sequence of Givens rotations that map W_z to W_a:
$$W_a \ = \ R_m(\tau_m) \ldots R_1(\tau_1)\ W_z \ .$$
Then the inverse mapping is obtained by reversing the sequence of rotations with the negative of the angles:
$$R(\boldsymbol{\tau}) \ = \ R_1(-\tau_1) \ldots R_m(-\tau_m) \ , \quad W_z = R(\boldsymbol{\tau}) W_a \ .$$

We made use of Givens rotations to interpolate projection frames. By comparison, the original grand tour implementation proposed in Asimov (1985), called "torus method," makes use of Givens decompositions in a somewhat different way: Asimov parametrizes the Stiefel manifold $V_{2,p}$ of 2-frames with angles $\boldsymbol{\tau} = (\tau_1, \tau_2, ...)$ provided by Givens rotations, and he devises infinite and uniformly distributed paths on the space (the "torus") of angles $\boldsymbol{\tau}$. The mapping of these paths to $V_{2,p}$ results in dense paths of frames, which therefore satisfy the definition of a grand tour. The non-uniformity problem mentioned at the end of the introduction stems from this mapping: The path of frames is not uniformly distributed, although its pre-image in the space of angles is. The non-uniformity may cause the tour to spend more time in some parts of the Stiefel manifold than in others. It would be of interest to better understand the mapping from the torus of angles to the manifold of frames. In particular, it would be interesting to know which of the many ways of constructing Givens decompositions lead to mappings with the best uniformity properties.

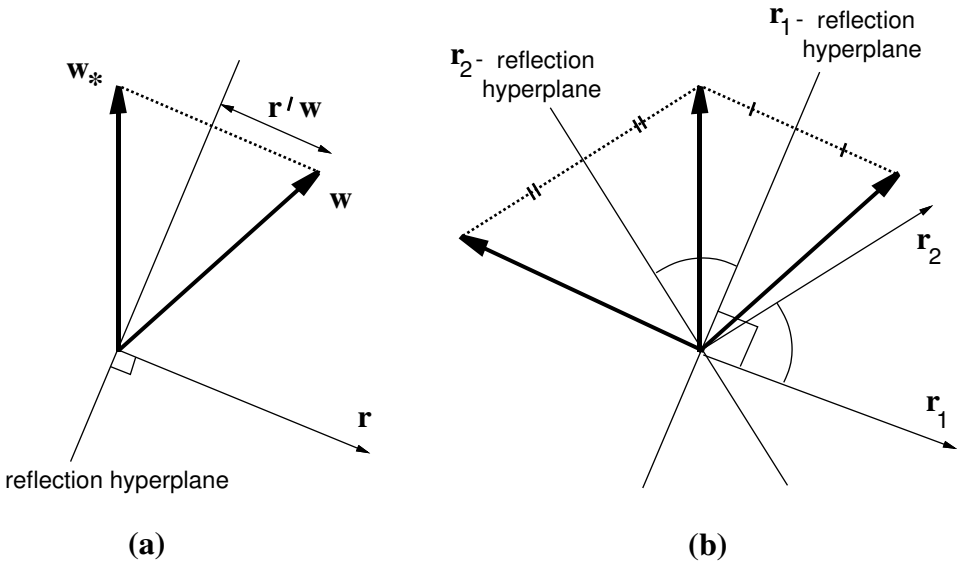

Figure 5: *(a) Reflection of a vector; (b) composition of two reflections to yield a planar rotation.*

4.3 Householder paths

This interpolation method is based on reflections on hyperplanes, also called "Householder transformations". See, for example, Golub and Van Loan (1983), Section 3.3. A reflection at the hyperplane with normal unit vector $\boldsymbol{r}$ is given by

$$H \;=\; I - 2 \cdot \boldsymbol{r}\boldsymbol{r}^T \;.$$

The usefulness of reflections stems from the following:

Facts:

1. *Any two distinct vectors of equal length, $\|\boldsymbol{w}\| = \|\boldsymbol{w}_*\|$, $\boldsymbol{w} \neq \boldsymbol{w}_*$, can be mapped onto each other by a uniquely determined reflection H with*

$$\boldsymbol{r} = (\boldsymbol{w} - \boldsymbol{w}_*)/\|\boldsymbol{w} - \boldsymbol{w}_*\| \;.$$

2. *Any vector orthogonal to $\boldsymbol{r}$ is fixed under the reflection.*

3. *The composition of two reflections H_1 and H_2 with vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, respectively, is a planar rotation in the plane spanned by $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, with an angle double the angle between $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$.*

See Figure 5 for a depiction. We illustrate the technique for $d = 2$. We use a first reflection H_1 to map $\boldsymbol{f}_{a,1}$ to $\boldsymbol{f}_{z,1}$. Subsequently, we use a second reflection H_2 to map

$H_1 \boldsymbol{f}_{a,2}$ to $\boldsymbol{f}_{z,2}$. Under H_2, the vector $H_1 \boldsymbol{f}_{a,1}$ is left fixed because both $H_1 \boldsymbol{f}_{a,2}$ and $\boldsymbol{f}_{z,2}$, and hence their difference, are orthogonal to $H_1 \boldsymbol{f}_{a,1} = \boldsymbol{f}_{z,1}$. Thus

$$F_z \;=\; H_2 H_1 F_a \;,$$

where $H_2 H_1$ is a planar rotation according to the third fact above. The computational details are somewhat more involved than in numerical analysis applications because we must make sure that we end up with exactly two reflections that amount to a planar rotation:

Path Construction (for $d = 2$):

1. Find a first reflection vector $\boldsymbol{r}_1$ such that $H_1 \boldsymbol{f}_{a,1} = \boldsymbol{f}_{z,1}$: If $\boldsymbol{f}_{a,1} \neq \boldsymbol{f}_{z,1}$, use $\boldsymbol{r}_1 = (\boldsymbol{f}_{a,1} - \boldsymbol{f}_{z,1})/\|\boldsymbol{f}_{a,1} - \boldsymbol{f}_{z,1}\|$, and $\boldsymbol{r}_1 \perp \boldsymbol{f}_{a,1}$ otherwise.

2. Map $\boldsymbol{f}_{a,2}$ with this reflection: $\boldsymbol{f}_{a,2+} = H_1 \boldsymbol{f}_{a,2} = \boldsymbol{f}_{a,2} - 2 \cdot (\boldsymbol{f}_{a,2}^T \boldsymbol{r}_1) \boldsymbol{r}_1$.

3. Find a second reflection vector $\boldsymbol{r}_2$ such that $H_2 \boldsymbol{f}_{a,2+} = \boldsymbol{f}_{z,2}$: If $\boldsymbol{f}_{a,2+} \neq \boldsymbol{f}_{z,2}$, use $\boldsymbol{r}_2 = (\boldsymbol{f}_{a,2+} - \boldsymbol{f}_{z,2})/\|\boldsymbol{f}_{a,2+} - \boldsymbol{f}_{z,2}\|$, and $\boldsymbol{r}_2 \perp \boldsymbol{f}_{a,2+}$, $\perp \boldsymbol{f}_{z,2}$ otherwise.

4. Form a preprojection basis B: Orthonormalize $\boldsymbol{r}_2$ with regard to $\boldsymbol{r}_1$ to yield $\boldsymbol{r}_*$; expand $(\boldsymbol{r}_1, \boldsymbol{r}_*)$ to an orthonormal basis of $\mathrm{span}(F_a, F_z)$.

5. Rotation: $R_{12}(\tau)$ with $\tau = 2\cos^{-1}(\boldsymbol{r}_1^T \boldsymbol{r}_2)$.

The generalization of Householder paths to d-frames for $d > 2$ is quite obvious for d even: The process generates d reflections that can be bundled up into $d/2$ planar rotations. For d odd, some precautions are in order: If $\mathrm{span}(F_a) \neq \mathrm{span}(F_z)$, one has to introduce one additional dummy reflection that leaves F_z fixed, using an $\boldsymbol{r}_{d+1}$ in $\mathrm{span}(F_a, F_z)$ orthogonal to F_z; if $\mathrm{span}(F_a) = \mathrm{span}(F_z)$, the last reflection was not necessary because $H_{d-1} ... H_1 \boldsymbol{f}_{a,d} = \pm \boldsymbol{f}_{z,d}$ automatically, hence $(d-1)/2$ rotations result. If the spans are identical, the frames have to have the same orientation in order to allow continuous interpolation; hence it may be necessary to change the sign of $\boldsymbol{f}_{z,d}$.

A peculiar aspect of the Householder method is that it uses fewer planar rotations than any of the other methods; as we have seen it transports 2-frames onto each other with a single planar rotation. Yet it does not produce paths that are optimal in any sense that we know of. It would be of interest to better understand the geometry of the Householder method and possibly produce criteria for which it is optimal. For example, we do not even know whether Householder paths are geodesic for one of the invariant metrics mentioned in Section 2.3.

5 Conclusions

The goal of this paper was to give an algorithmic framework for dynamic projections based on the interpolation of pairs of projections. The notion of steering from target projection to target projection makes for a flexible environment in which grand tours, interactive projection pursuit and manual projection control can be nicely embedded. Finally, we proposed several numerical techniques for implementing interpolating paths of projections.

References

[1] Asimov, D. (1985), "The grand tour: a tool for viewing multidimensional data," *SIAM J. Sci. Statist. Computing* **6** 1, 128–143.

[2] Asimov, D., and Buja, A. (1994), "The grand tour via geodesic interpolation of 2-frames," in *Visual Data Exploration and Analysis, Symposium on Electronic Imaging Science and Technology*, IS&T/SPIE (Soc. for Imaging Sci. and Technology/Internat. Soc. for Optical Engineering).

[3] Bjorck, A., and Golub, G. H. (1973), "Numerical methods for computing angles between linear subspaces," *Mathematics of Computation* **27** 123, 579–594.

[4] Buja, A., and Asimov, D. (1986), "Grand tour methods: an outline," *Computer Science and Statistics: Proc. of the 17th Symp. on the Interface between Comput. Sci. and Statist.*, Amsterdam: Elsevier, 63–67.

[5] Buja, A., Asimov, D., Hurley, C., and McDonald, J. A. (1988), "Elements of a viewing pipeline for data analysis," in *Dynamic Graphics for Statistics*, eds. W. S. Cleveland and M. E. McGill, Belmont, CA: Wadsworth, 277–308.

[6] Buja, A., Hurley, C., and McDonald, J. A. (1986), "A data viewer for multivariate data," *Computer Science and Statistics: Proc. of the 18th Symp. on the Interface between Comput. Sci. and Statist.*, Amsterdam: Elsevier.

[7] Buja, A., Cook, D., and Swayne, D. F. (1996), "Interactive high-dimensional data visualization," *Journal of Computational and Graphical Statistics* **5**, 78–99.

[8] Buja, A., Cook, D, Asimov, D., Hurley, C.. (2004), "Theory of Dynamic Projections in High-Dimensional Data Visualization," submitted; can be downloaded from www-stat.wharton.upenn.edu/~buja/

[9] Carr, D. B., Wegman, E. J., Luo, Q. (1996), "ExploreN: Design considerations past and present," Technical Report 129, Center for Computational Statistics, George Mason University, Fairfax, VA 22030.

[10] Conway, J. H., Hardin, R. H., and Sloane, N. J. A. (1996), "Packing lines, planes, etc.: Packings in Grassmannian spaces," *Journal of Experimental Mathematics* **5**, 139–159.

[11] Cook, D., and Buja, A. (1997), "Manual controls for high-dimensional data projections," *Journal of Computational and Graphical Statistics* **6**, 464–480 (1997).

[12] Cook, D., Buja, A., Cabrera, J., and Hurley, H. (1995), "Grand tour and projection pursuit," *J. of Computational and Graphical Statistics* **2** 3, 225–250.

[13] Cook, D. R., and Weisberg, S. (1994), *An Introduction to Regression Graphics*, New York: Wiley.

[14] Duffin, K. L., and Barrett, W. A. (1994), "Spiders: a new user interface for rotation and visualization of N-dimensional point sets," in *Proceedings Visualization '94*, IEEE Computer Society Press, Los Alamitos, California, 205–211.

[15] Furnas G. W., and Buja A. (1994), "Prosection Views: Dimensional Inference through Sections and Projections," *Journal of Computational and Graphical Statistics*, **3**, 323–385.

[16] Golub, G. H., and Van Loan, C. F. (1983), *Matrix Computations*, second edition, Baltimore, Maryland: The Johns Hopkins University Press.

[17] Halmos, P. R. (1958), *Finite-Dimensional Vector Spaces*, New York: Springer.

[18] Halmos, P. R. (1970), "Finite-Dimensional Hilbert Spaces," *The American Mathematical Monthly,* **77** 5, 457–464.

[19] Hurley, C., and Buja, A. (1990), "Analyzing high-dimensional data with motion graphics," *SIAM Journal on Scientific and Statistical Computing*, **11** 6, 1193–1211.

[20] McDonald, J. A. (1982), "Orion I: Interactive graphics for data analysis," in *Dynamic Graphics for Statistics*, eds. W. S. Cleveland and M. E. McGill, Belmont, CA: Wadsworth.

[21] Swayne, D. F., Cook, D., and Buja, A. (1998), "XGobi: Interactive Dynamic Data Visualization in the X Window System," *Journal of Computational and Graphical Statistics*, **7** 1, 113–130.

[22] Swayne, D.F., Buja, A., Temple-Lang, D. (2003), "Exploratory Visual Analysis of Graphs in GGobi," Proceedings of the *Third Annual Workshop on Distributed Statistical Computing* (DSC 2003), Vienna.

[23] Swayne, D.F., Temple-Lang, D., Buja, A., and Cook, D. (2002), "GGobi: Evolving from XGobi into an Extensible Framework for Interactive Data Visualization," *Journal of Computational Statistics and Data Analysis*.

[24] Symanzik, J., Wegman, E., Braverman, A., and Luo, Q. (2002), "New applications of the image grand tour," *Computing Science and Statistics*, **34**, 500–512.

[25] Tierney, L. (1990), *Lisp-Stat*, New York, NY: Wiley.

[26] Tukey, J. W. (1987), "Comment on 'Dynamic graphics for data analysis' by Becker et al.," *Statistical Science*, **2** 355-395; also in *Dynamic Graphics for Statistics*, eds. W. S. Cleveland and M. E. McGill; Belmont, CA: Wadsworth.

[27] Wegman, E. J. (1991), "The grand tour in k-dimensions," *Computing Science and Statistics: Proceedings of the 22nd Symposium on the Interface*, 127–136.

[28] Wegman, E. J. (2003), "Visual data mining," *Statistics in Medicine*, **22**, 1383–1397, plus 10 color plates.

[29] Wegman, E. J., and Carr, D. B. (1993), "Statistical graphics and visualization," in *Handbook of Statistics 9: Computational Statistics*, 857–958, ed. C. R. Rao; Amsterdam: Elsevier.

[30] Wegman, E. J. and Shen J. (1993), "Three-dimensional Andrews plots and the grand tour," *Computing Science and Statistics*, **25**, 284–288.

[31] Wegman, E. J., Poston, W. L., and Solka, J. L. (1998), "Image grand tour," *Automatic Target Recognition VIII - Proceedings of SPIE*, 3371, 286–294.

[32] Wegman, E. J., and Solka, J. L. (2002), "On some mathematics for visualizing high dimensional data," *Sanhkya (A)*, **64** (2), 429–452.