Spiders: A New User Interface for Rotation and Visualization of N-dimensional Point Sets

Kirk L. Duffin
Brigham Young University
kirkl@python.cs.byu.edu

William A. Barrett
Brigham Young University

Abstract

We present a new method for creating n-dimensional rotation matrices from manipulating the projections of n-dimensional data coordinate axes onto a viewing plane. A user interface for n-dimensional rotation is implemented. The interface is shown to have no rotational hysteresis.

1 Introduction

Many techniques for visualizing n-dimensional data sets separate the data into its component dimensions, allowing the user to look at various coordinate combinations in a way that hopefully brings understanding. These methods do well at avoiding the traditional projection to two dimensions that hides data. However the data relationships are not immediately intuitive to our brains, which are used to transforming large amounts of information from three dimensional projections down to two.

On the other hand, projection of n-dimensional information down to two may be slightly more intuitive, but suffers from the curse of data hiding due to projection. Moving the data in n-space, by predetermined motion or direct manipulation can help solve this problem.

Asimov's "grand tour"[Asi85] made it possible to step through all possible projections of an n-dimensional data set onto two dimensions in a useful manner. Hurley and Buja introduced a means of creating "guided tours" of the data by allowing the user to create two disparate projection plane orientations and interpolate between them[Hur88]. A good method of interpolation is to create a n-dimensional rotation between the two orientations and sample along the rotation angle[BA86]. Subsequent data rotation tools, while similar, have retained this interpolation approach for creating smooth motion in the projected data[YR91, SC90].

Here we present a new technique for creating n-dimensional rotations from information projected onto the viewing plane. From this technique we develop an interface for interactively rotating n-dimensional point sets. User control over the rotation sequence is fine enough that no direct interpolation between projections is needed. Section 2 will review some of the important principles from matrix algebra. Section 3 will develop the main algorithm for creating n-dimensional rotation matrices from manipulation of data projections in the viewing plane. Section 4 will discuss some of the implementation aspects of the algorithm and present an implementation of an interactive n-dimensional rotation interface that is free from hysteresis effects. Section 5 will demonstrate the manipulation of two 5-dimensional data sets using the interface, and section 6 will point out some possible areas of refinement for the interface.

2 Background

2.1 Notation

In this paper we will hold to an extension of the notation used in most of the computer graphics literature: a n-dimensional point is represented by a n-dimensional row vector and is post-multiplied by any transformation matrices. The vector composed of all zeros except for a 1 in position i will be denoted $\mathbf{e}_i$.

2.2 Coordinate Frames

We represent a n-dimensional data set as a set of points in an n-dimensional Euclidian space $\mathbb{R}^n$. There are two ways of investigating the projection of a set of n-dimensional points onto a 2-dimensional viewing plane. In the first, the coordinate system of the data and the coordinate system of the viewing space coincide. A viewing plane is arbitrarily placed in the viewing space and the data is projected onto the plane. The second approach to projecting n-dimensional data onto a viewing plane moves the coordinate system of the data with respect to the coordinate system of the viewing space. In this latter approach the viewing plane remains fixed.

Because a rotation leaves the coordinate system origin invariant, it is possible to focus on the rotation as a transformation of a vector from the origin to the data point. This allows the creation of **coordinate frames**, a cluster of unit vectors that point down the positive principal axes of the underlying coordinate system.

Using coordinate frames gives us some powerful tools[Piq90]. If we start with an untransformed data coordinate frame, multiplying each axis vector $\mathbf{e}_i$ in turn by the rotation matrix, it can be seen that the new position in view space of the axis is given by row i of the rotation matrix.

A corollary to this fact is that if we specify the new position of the axis vectors such that they remain orthonormal then the new positions define the rows of

the rotation matrix[1] $\mathbf{R}$.

2.3 Orthogonal Projections in n Dimensions

We define the orthogonal projection of an n dimensional point onto a subspace of lower dimension (the viewing subspace) as the point in the subspace closest to the data point. If $\mathbf{b}_1 \cdots \mathbf{b}_m, m \leq n$ are basis vectors of the viewing subspace, then the projection $\mathbf{x}_{proj}$ of a data point $\mathbf{x}$ is defined by

$$\mathbf{x}_{proj} = \sum_{i=1}^{m} \mathbf{x} \cdot \mathbf{b}_i. \tag{1}$$

If the $\mathbf{b}$ are equivalent to the standard basis vectors $\mathbf{e}_i$ then the projection of $\mathbf{x}$ onto $\mathbf{b}_i$ is simply the i-th coordinate of $\mathbf{x}$.

3 Arbitrary Rotations in n Dimensions

In three dimensions, rotations are commonly specified in terms of an angle about an arbitrary axis. However, it is more correct to think of rotation as taking place in a plane embedded in the space[Nol67]. In 3-dimensional rotations, this plane is the plane perpendicular to the axis of rotation. In more than three dimensions, the idea of rotation about an axis goes awry because there are an infinite number of axes that are perpendicular to any given plane. But as long as a plane in the space is specified along with a center of rotation in the plane, the rotation is uniquely defined.

The simplest rotation to describe in n-dimensional space occurs in the plane formed by any two coordinate axes. The rotation matrix $\mathbf{R}_{ab}(\theta)$ for the rotation of axis $\mathbf{x}_a$ in the direction of $\mathbf{x}_b$ by the angle θ is

$$\mathbf{R}_{ab}(\theta) = \left\{ r_{ij} \;\middle|\; \begin{array}{rcrl} r_{ii} & = & 1 & i \neq a, i \neq b \\ r_{aa} & = & \cos\theta & \\ r_{bb} & = & \cos\theta & \\ r_{ab} & = & -\sin\theta & \\ r_{ba} & = & \sin\theta & \\ r_{ij} & = & 0 & \text{elsewhere} \end{array} \right\}. \tag{2}$$

That is, $\mathbf{R}_{ab}(\theta)$ is an identity matrix except for the entries at the intersection of rows a and b and columns a and b. Since there are $\binom{n}{2}$ principal axes planes, n-dimensional rotations are built up as the composition of specified rotations in each of the principal planes. This composition is accomplished by multiplying the corresponding rotation matrices together.

Our goal is to provide an intuitive means of specifying an n-dimensional interface, hopefully in a concise graphical manner. The key to our approach is in the observation that if an axis is not contained in the viewing plane nor is perpendicular to the viewing plane, then the axis and its projection onto the viewing plane define another plane in which rotation can occur. Moreover, by manipulating the projection of an axis, it is possible to rotate the axis in the rotation plane such that the axis remains consistent with its projection. Figure 1 illustrates this observation for $n = 3$.

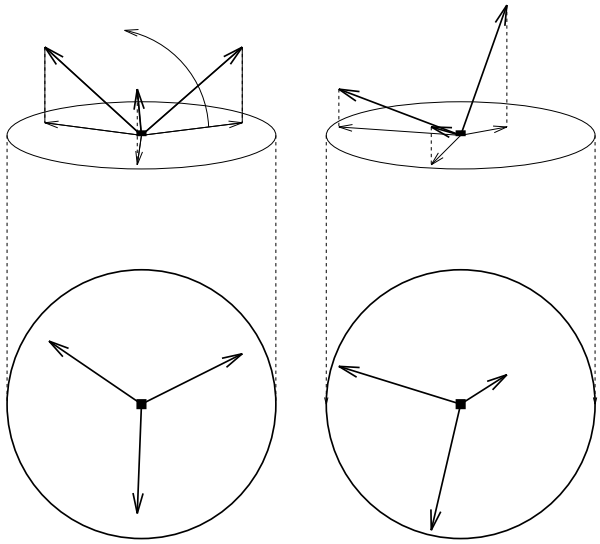

Figure 1: A 3-dimensional coordinate frame before rotation (l) and after rotation (r). The rotation plane is defined by the rightmost data axis in each diagram and its projection. The circle at the bottom of each diagram shows the projection of the data coordinate axes onto the viewing plane.

3.1 Rotation in the Plane

The problem here is to rotate the selected axis $\mathbf{x}_i$ by an unknown angle θ to its new position $\mathbf{x}'_i$. All that is known are the magnitudes of the projections of the axis.

Let $\mathbf{x}_i$ be a unit vector representing the positive direction of the i-th axis of the data coordinate system embedded in the n-dimensional viewing coordinate system. The projection of $\mathbf{x}_i$ onto the viewing plane is denoted $\mathbf{x}_{i_{proj}}$. The position and projection of the axis after rotation are denoted $\mathbf{x}'_i$ and $\mathbf{x}'_{i_{proj}}$ respectively. See figure 2.

In the rotation plane, $\mathbf{x}_i$ can be decomposed into two vectors, $\mathbf{x}_{i_{proj}}$, and a component orthogonal to the viewing plane, $\mathbf{x}_{i_\perp}$ such that $\mathbf{x}_{i_\perp} = \mathbf{x}_i - \mathbf{x}_{i_{proj}}$. These two vectors set up an orthogonal coordinate system in the rotation plane. Now $\mathbf{x}_i$ can be represented by the coordinates $(m_{i_{proj}}, m_{i_\perp})$ where $m_{i_{proj}} = \|\mathbf{x}_{i_{proj}}\|$ and $m_{i_\perp} = \|\mathbf{x}_{i_\perp}\|$.

Since $\mathbf{x}_i$ and $\mathbf{x}'_i$ are unit vectors, given $m'_{i_{proj}}$, the magnitude of the new projected component can be determined, namely $m'_{i_\perp} = \sqrt{1 - m'^{\,2}_{i_{proj}}}$. Conse-

[1] Actually, this is not quite true. The negation of a data axis is also allowed in this definition which corresponds to a reflection of the data about that axis. However, the algorithm presented here will not produce reflections.

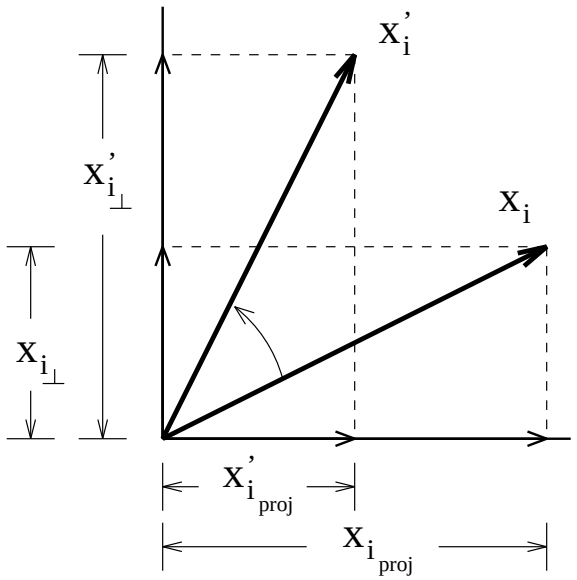

Figure 2: Rotation in the plane defined by $\mathbf{x}_i$ and its projection on the viewing plane. The data coordinate axis vector $\mathbf{x}_i$ is rotated to $\mathbf{x}'_i$.

quently, the parameters for rotation in the plane are

$$\cos\theta = \mathbf{x}_i \cdot \mathbf{x}'_i = m_{i_{proj}} m'_{i_{proj}} + m_{i_\perp} m'_{i_\perp} \quad (3)$$

$$\sin\theta = \|\mathbf{x}_i \times \mathbf{x}'_i\| = m_{i_{proj}} m'_{i_\perp} - m_{i_\perp} m'_{i_{proj}}. \quad (4)$$

Thus any vector $\mathbf{v}$ in the plane can be rotated using the standard rotation equations

$$\mathbf{v}' = \left(\frac{\mathbf{v} \cdot \mathbf{x}_{i_{proj}}}{\|\mathbf{x}_{i_{proj}}\|}, \frac{\mathbf{v} \cdot \mathbf{x}_{i_\perp}}{\|\mathbf{x}_{i_\perp}\|} \right) \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \quad (5)$$

More importantly,

$$\mathbf{x}'_i = m'_{i_\perp} \frac{\mathbf{x}_{i_\perp}}{\|\mathbf{x}_{i_\perp}\|} + m'_{i_{proj}} \frac{\mathbf{x}_{i_{proj}}}{\|\mathbf{x}_{i_{proj}}\|} \quad (6)$$

To determine the n-dimensional rotation matrix $\mathbf{R}$, all that remains is to find the new positions of each axis vector. This is accomplished by decomposing each data space coordinate axis vector into three components: a vector orthogonal to the rotation plane, and two vector components in the rotation plane. These last two vectors are the projection of the data axis vector onto $\mathbf{x}_{i_{proj}}$ and $\mathbf{x}_{i_\perp}$ respectively. The rotation is calculated for the rotation plane components and the results added to the orthogonal vector component. This gives the rotated position of the axis vector.

Let a and b be the coordinates of data axis $\mathbf{x}_j$ projected onto the rotation plane, i.e.

$$a = \mathbf{x}_j proj \mathbf{x}_{i_{proj}} = \frac{\mathbf{x}_j \cdot \mathbf{x}_{i_{proj}}}{\|\mathbf{x}_{i_{proj}}\|} \quad (7)$$

$$b = \mathbf{x}_j proj \mathbf{x}_{i_\perp} = \frac{\mathbf{x}_j \cdot \mathbf{x}_{i_\perp}}{\|\mathbf{x}_{i_\perp}\|}. \quad (8)$$

Let $\mathbf{x}_{j_{orth}}$ be the orthogonal component of $\mathbf{x}_j$ with respect to the rotation plane. Then

$$\mathbf{x}_{j_{orth}} = \mathbf{x}_j - a \frac{\mathbf{x}_{i_{proj}}}{\|\mathbf{x}_{i_{proj}}\|} - b \frac{\mathbf{x}_{i_\perp}}{\|\mathbf{x}_{i_\perp}\|}. \quad (9)$$

After rotation, the new position of the data axis vector $\mathbf{x}'_j$ can be expressed

$$\begin{aligned} \mathbf{x}'_j &= \mathbf{x}_{j_{orth}} + (a\cos\theta - b\sin\theta) \frac{\mathbf{x}_{i_{proj}}}{\|\mathbf{x}_{i_{proj}}\|} \\ &\quad + (a\sin\theta + b\cos\theta) \frac{\mathbf{x}_{i_\perp}}{\|\mathbf{x}_{i_\perp}\|}. \end{aligned} \quad (10)$$

Substituting (9) into (10) and simplifying results in

$$\begin{aligned} \mathbf{x}'_j &= \mathbf{x}_j + (a(\cos\theta - 1) - b\sin\theta) \frac{\mathbf{x}_{i_{proj}}}{\|\mathbf{x}_{i_{proj}}\|} \\ &\quad + (b(\cos\theta - 1) + a\sin\theta) \frac{\mathbf{x}_{i_\perp}}{\|\mathbf{x}_{i_\perp}\|}. \end{aligned} \quad (11)$$

3.2 Algorithm

The foregoing development gives us the following algorithm for creating an n-dimensional rotation matrix.

Input:

- $\mathbf{R}$ — the current rotation matrix. The rows of this matrix are the axis vectors of the data coordinate system. The elements of $\mathbf{R}$ are denoted r_{ij}.
- i — the index of the data coordinate axis that determines the plane of rotation.
- m'_{proj} — the desired magnitude of the projected component of the selected data axis.
- $axis_1$, $axis_2$ — the viewing space axes defining the viewing plane.

Output:

- $\mathbf{R}'$ — the new rotation matrix describing the transformation from data coordinate space to viewing coordinate space.

Variables:

- m_{proj} — the current magnitude of the projected component of the selected data axis.
- $m_\perp$ — the current magnitude of the orthogonal component of the selected data axis.
- $m'_\perp$ — the orthogonal component magnitude of the rotated data axis.
- cos, sin — the rotation parameters of the rotation.
- k_1, k_2, sum — intermediate values

Find magnitude of projected component of selected axis.

$sum \leftarrow 0$
for $(1 \leq \ell \leq 2)$
  $sum \leftarrow sum + r^2_{i\,axis_\ell}$
$m_{proj} = \sqrt{sum}$

Find component magnitude of selected axis perpendicular to viewing plane.

$sum \leftarrow 0$
for $(1 \leq \ell \leq n)$
  if $(\ell \neq axis_1$ and $\ell \neq axis_2)$
    $sum \leftarrow sum + r^2_{i\ell}$
$m_\perp = \sqrt{sum}$

$$m'_\perp = \sqrt{1 - {m'_{proj}}^2}$$

Calculate projection plane parameters.

$cos \leftarrow m_{proj} * m'_{proj} + m_\perp * m'_\perp$
$sin \leftarrow m_{proj} * m'_\perp - m_\perp * m'_{proj}$

Rotate each data space axis.

for $(1 \leq j \leq n)$
$sum \leftarrow 0$
  for $(1 \leq \ell \leq 2)$
    $sum \leftarrow sum + r_{j\,axis_\ell} * r_{i\,axis_\ell}$
  $a \leftarrow sum/m_{proj}$
  $sum \leftarrow 0$
  for $(1 \leq \ell \leq n)$
    if $(\ell \neq axis_1$ and $\ell \neq axis_2)$
      $sum \leftarrow sum + r_{i\ell} * r_{j\ell}$
  $b \leftarrow sum/m_\perp$
  $k_1 \leftarrow (a * (cos - 1) - b * sin)/m_{proj}$
  $k_2 \leftarrow (b * (cos - 1) + a * sin)/m_\perp$
  for $(1 \leq \ell \leq n)$
    if $(\ell = axis_1$ or $\ell = axis_2)$
      $r'_{j\ell} \leftarrow r_{j\ell} + k_1 * r_{i\ell}$
    else
      $r'_{j\ell} \leftarrow r_{j\ell} + k_2 * r_{i\ell}$

The simplified formulation of the main inner loop from (11) is justified by noting that if we limit the viewing plane to be one of the principal planes in the viewing coordinate system, then $\mathbf{x}_{i_{proj}}$ has non-zero components only along the axes specified by the viewing plane. Likewise, $\mathbf{x}_{i_\perp}$ will always have 0 coordinates in those two dimensions.

4 Implementation

4.1 Interface

We have used two approaches in applying the above formulas to the development of user interfaces for n-dimensional rotation. Each approach allows the user to select a data coordinate axis and drag the projected end of the axis in the viewing plane. From the path traversed in the viewing plane, a sequence of n-dimensional rotation matrices is created. The difference in the two approaches is in how the change in position of a selected projected axis is turned into a rotation matrix.

In the first approach $\mathbf{R}$ is composed of two rotations, the first occurs in the plane formed by axis $\mathbf{x}_i$ and its projection $\mathbf{x}_{i_{proj}}$. The amount of rotation is determined by the change in length of the projected axis. The second rotation occurs in the viewing plane and accounts for the change in projected orientation of $\mathbf{x}_{i_{proj}}$. Figure 3 illustrates for $n = 3$.

However, rotation in the projection plane provides no new visual information. In practice, the set of projected axes tends to spin wildly in the viewing plane. This in turn makes it difficult to adjust the relative positions of the projected axes.

The second approach to the creation of n-dimensional rotation matrices also decomposes $\mathbf{R}$ into two rotations. The first rotation rotates the selected axis $\mathbf{x}_i$ in the plane formed by itself and its original projection so that $\mathbf{x}_i$ is perpendicular to the viewing plane. The second rotation rotates $\mathbf{x}_i$ from its position perpendicular to the viewing plane to a position consistent with the projected position. (Figure 4).

4.2 Lack of Hysteresis

This latter approach to rotation possesses a nice theoretical quality. Let the rotation of $\mathbf{x}_i$ from its position on the viewing plane, $\mathbf{x}_{i_{proj}} = (u_j, v_j)$ to its new position $\mathbf{x}'_{i_{proj}} = (u_{j+1}, v_{j+1})$ be denoted ${}_j\mathbf{R}_{j+1}$ for any j. But this is the composition of two other matrices, ${}_j\mathbf{R}_{j+1} = {}_j\mathbf{P}\mathbf{Q}_{j+1}$ where ${}_j\mathbf{P}$ is the rotation of $\mathbf{x}_i$ to a position perpendicular to the viewing plane and $\mathbf{Q}_{j+1}$ is the rotation of $\mathbf{x}_i$ from the perpendicular space to its new position corresponding to $\mathbf{x}'_{i_{proj}}$.

Now if a user selects an axis $\mathbf{x}_i$ at position (u_0, v_0) on the viewing plane and drags the projected axis around the viewing plane, then the rotation matrix of this transformation is the composition of the rotation matrices of every point on the path of the dragged projected axis in the viewing plane, i.e.

$$ {}_0\mathbf{R}_m = {}_0\mathbf{P}\mathbf{Q}_1\,{}_1\mathbf{P}\mathbf{Q}_2 \cdots {}_j\mathbf{P}\mathbf{Q}_{j+1} \cdots {}_{m-1}\mathbf{P}\mathbf{Q}_m \quad (12)$$

for the path in the viewing plane of (u_0, v_0), $\cdots$, (u_m, v_m).

But rotating an axis perpendicular to the viewing plane and then rotating it back to the same position is an identity operation. This means that $\mathbf{Q}_j\,{}_j\mathbf{P} = I$. Consequently, (12) collapses to

$$ {}_0\mathbf{R}_m = {}_0\mathbf{P}\mathbf{Q}_m. \quad (13)$$

Thus dragging a projected axis with this method is a conservative operation. The rotation matrix resulting from dragging $\mathbf{x}_{i_{proj}} = (u_0, v_0)$ to its new position $\mathbf{x}'_{i_{proj}} = (u_m, v_m)$ is the same, regardless of the path taken from (u_0, v_0) to (u_m, v_m)[2].

This lack of hysteresis is a highly desirable property for interactive rotational interfaces for at least two reasons: First, the user can follow any path in the viewing plane when dragging a projected axis and be guaranteed of receiving the same rotation matrix, given the same start and end points of the drag. If

[2] As long as the path does not pass through the projection of the origin of the data coordinate viewing system onto the viewing plane.

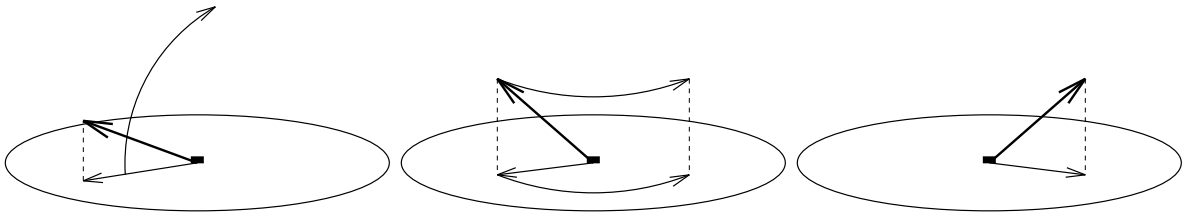

Figure 3: Repositioning a projected coordinate axis by 1) rotating for new projected coordinate axis length, and 2) rotating in the viewing plane for new projected coordinate axis orientation.

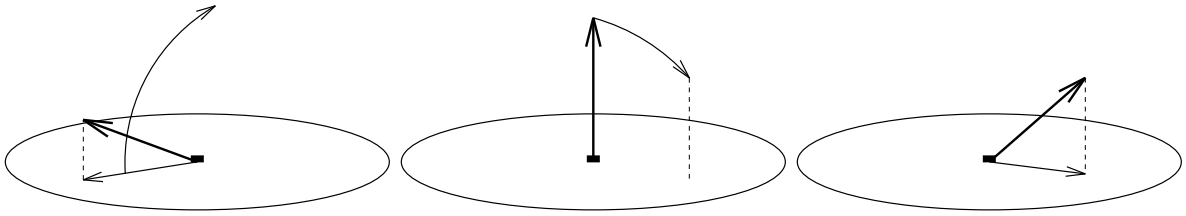

Figure 4: Repositioning a projected coordinate axis by 1) rotating axis perpendicular to viewing plane, and 2) rotating out of perpendicular space to new projected axis position.

the desired projected target is overshot or missed, the axis can be dragged back to the desired position. Secondly, the interface need not process every point in the path of the dragged projected axis in order to maintain consistent interface operation. If the data is being replotted as the axis is dragged, then a rotation matrix need be created only from the current viewing plane position. Any other positions traversed since the last plotting step can be discarded, giving a great computational savings if the data set is large.

The unit quaternions also share this lack of hysteresis, which has generated significant interest in their use in 3-dimensional rotation interfaces[Sho92]. Now this property can be extended to n-dimensional rotations as well.

4.3 Orthonormality

As presented, this algorithm is highly dependent on the fact that the axis vectors are orthonormal. In practice, as numerical error creeps in, the rotation matrix $\mathbf{R}$ ceases to be orthogonal. This is a standard problem in 3-d interfaces where the rotation matrix is occasionally re-orthogonalized. Our experience has been that renormalizing the rows of the rotation matrix is sufficient to maintain orthogonality. Without renormalization, numerical error quickly dominates, making $\mathbf{R}$ useless.

Actually, nothing in the derivation of the algorithm depends on the mutual orthogonality of the coordinate axis vectors. Therefore, the algorithm given above will properly transform any set of vectors through a rotation specified by a n-dimensional vector and its projection. But in such a case, the resulting vectors can not be used to form the new rotation matrix.

4.4 Boundary Conditions

Because the algorithm decomposes every n-space vector into two rotation plane components, it is necessary that the axis $\mathbf{x}_i$ that determines the rotation plane be distinct from its projection onto the viewing plane. Consequently, special measures must be taken when $\mathbf{x}_i$ lies in the viewing plane or is perpendicular to the viewing plane. In practice, due to discretization error in the interface, conditions when $\mathbf{x}_i$ is close to the viewing plane or close to the perpendicular must also be considered.

In our implementation, when an axis projection is dragged within a small distance of the center of the viewing plane, the axis snaps perpendicular to the viewing plane and stays there. When a user wishes to drag one axis (of possibly several) out of the space perpendicular to the viewing plane, she clicks the mouse on the center of the projected coordinate frame. A text menu offers a selection of the available perpendicular axes. After a selection is made, a point on the viewing plane is selected, and the axis is rotated out to this position. From there the axis can be dragged like any others visible on the viewing plane.

5 Application

In our work in the Brigham Young University Computer Vision Laboratory we have implemented this interface to help visualize images and color gamuts as 5-dimensional point sets. Each pixel in a full color image is given five spatial coordinates: x, y, red, green, and blue. Each of these data points is also given a color corresponding to its red, green, and blue

components. This is done for convenience only and is not necessary for the functioning of the interface. The mean of the data set is subtracted from all points so that rotation will occur about the center of the data set.

The orthogonal projection of the data set is kept separate from the rotational interface, which has acquired the appellation of a"spider." This is due to the appearance of many moving "legs" on the viewing plane when many coordinate axes are simultaneously visible.

Our combining the projected axes into one figure is in direct contrast to Hurley's *data viewer*[HB90], which assigns each axis its own interface item. Our experience seems to indicate that combining the axes into a single figure is acceptable when using relatively low dimension data sets. However, we have implemented the spiders with the facility to display an arbitrary subset of the full data axis complement. We have also used the powerful concept of linking demonstrated by Buja, McDonald, et al[BMMS91] to link several spiders simultaneously to a single data set.

Figure 5 shows an image undergoing 5D rotation. At first only the x and y components are visible. Then the red axis is dragged out of the space perpendicular to the viewing plane. Because of the correspondence between the color attributes and the spatial coordinates, all of the points with high red values appear to move in the direction of the projected red axis. Note that as the red axis is brought out slightly, a pseudo 3D effect occurs. Next the green axis is dragged out and the x axis pushed back into the perpendicular space. Finally, the blue axis is brought out, the y axis pushed in, and the three remaining color axes arranged evenly in the projection plane. The points in the data set realign themselves into a pattern reminiscent of a color wheel.

6 Conclusion

We have demonstrated a new method called "spiders" for interactively rotating n-dimensional point sets. The technique provides n-dimensional rotation matrices solely from information about the current data coordinate system and its projection onto the viewing plane. The interface has no rotational hysteresis, similar to the more robust 3D interfaces used today.

The spiders are not without problems. They do suffer from the "curse of projection" and data hiding with dense sets associated with all projective techniques. And like other visualization methods, as more dimensions are added to the system, the incremental return in understanding decreases. Nevertheless, we feel that the interactive nature of this technique provides a powerful tool to help understand the universe of data around us.

References

[Asi85] Daniel Asimov. The grand tour: A tool for viewing multidimensional data. *SIAM Journal on Scientific and Statistical Computing*, 6(1):128–143, January 1985.

[BA86] Andreas Buja and Daniel Asimov. Grand tour methods: An outline. In *Proceedings of the 18th Symposium on the Interface*, pages 63–67. American Statistical Association, 1986.

[BMMS91] Andreas Buja, John Alan McDonald, John Michalak, and Werner Stuetzle. Interactive data visualization using focusing and linking. In *IEEE Conference on Visualization*, pages 156–163, 1991.

[HB90] Catherine Hurley and Andreas Buja. Analyzing high-dimensional data with motion graphics. *SIAM Journal on Scientific and Statistical Computing*, 11(6):1193–1211, November 1990.

[Hur88] Catherine Hurley. A demonstration of the data viewer. In *Proceedings of the 20th Symposium on the Interface*, pages 108–113. American Statistical Association, 1988.

[Nol67] A. Michael Noll. A computer technique for displaying n-dimensional hyperobjects. *Communications of the ACM*, 10(8):469–473, August 1967.

[Piq90] Michael E. Pique. Rotation tools. In Andrew S. Glassner, editor, *Graphics Gems*, pages 465–469. Academic Press, 1990.

[SC90] Deborah F. Swayne and Dianne Cook. Xgobi: A dynamic graphics program implemented in x with a link to s. In *Proceedings of the 22nd Symposium on the Interface*, pages 544–547. American Statistical Association, 1990.

[Sho92] Ken Shoemake. Arcball: A user interface for specifying three-dimensional orientation using a mouse. In *Proceedings of Graphics Interface '92*. Morgan Kaufmann, 1992.

[YR91] Forrest W. Young and Penny Rheingans. Visualizing structure in high-dimensional multivariate data. *IBM Journal of Research and Development*, 35(1):97–107, 1991.