Techniques for Non-Linear Magnification Transformations

T. Alan Keahey*
Department of Computer Science
Indiana University
215 Lindley Hall
Indiana University
Bloomington, IN 47405
Phone: (812) 855-3609
Fax: (812) 855-4829
tkeahey@cs.indiana.edu

Edward L. Robertson
Department of Computer Science
Indiana University
edrbtsn@cs.indiana.edu

March 29, 1996

Abstract

This paper presents some methods for implementing efficient and general non-linear magnification transformations. Techniques are provided for combining linear and non-linear magnifications, for constraining the domain of magnifications, and for combining multiple transformations. In addition piecewise linear methods are introduced which allow greater efficiency and expressiveness than their continuous counterparts.

Keywords

Distortion, Magnification, Fisheye, Hyperbolic, Information Visualization, Piecewise Transformation

*This work was supported by US Dept. of Education award number P200A502367.

1 Introduction

Since the introduction of the video display terminal, computer users have had to deal with the problem of limited screen space. This problem is becoming increasingly significant as modern computer systems and applications become more graphics oriented. Although current screen sizes of 1000×800 pixels are increasingly common, they are still not adequate to completely display many complex visual scenes.

Conversely, an argument can also be made that screen sizes have become too large. As resolution increases, more graphical information can be displayed on a single frame, oftentimes beyond an user's perceptual limit. Well designed layouts can help in many cases, although a general purpose mechanism for allowing the user to explicitly enhance visualization of some area(s) of the screen would facilitate better focus on items of interest.

Magnification is one tool that can be brought to bear on both of these problems. This paper will discuss general, efficient techniques for performing magnification. All of the examples discussed in this paper were obtained from the FAD magnification toolkit [1] being developed by the author, which allows highly interactive construction and application of magnifications, as well as a programmer's interface to a library of transformation routines. All transformations provide interactive frame rates as the user moves the magnification over a domain and adjusts the magnification parameters [2].

2 Magnification Transformations

The available literature on normal and distortion-based magnification abounds with a full range of terminologies to describe the techniques and characteristics of magnification/zoom/distortion/warp/etc. Where possible we follow the lead of Leung and Apperley [12], whose review and taxonomy of these techniques provides a basis for discussion. In particular, Leung and Apperley make the distinction between *transformation* functions and *magnification* functions, and show the relationship between them. Figure 1 shows a simple linear *transformation* function on the left, and the corresponding *magnification* function on the right. The magnification function is the derivative of the transformation function. It will be shown in this paper that it is useful to be able to translate between these two functions. Related work illustrating a technique for visualizing these transformations has been done by Furnas and Bederson [6].

2.1 Linear Transformations

The most familiar magnification technique involves the application of linear transformation functions as shown in Figure 1. The result of these transformations is a constant level of magnification

[1] FAD = Future Acronym in Development

[2] A video of these capabilities is in the works, but not available at the time of this paper's submission.

across the domain, very much similar to what someone would perceive through an ordinary magnifying glass. Such techniques have an advantage of providing a close analogy to familiar experiences for the user, however certain limitations are inherent with this approach:

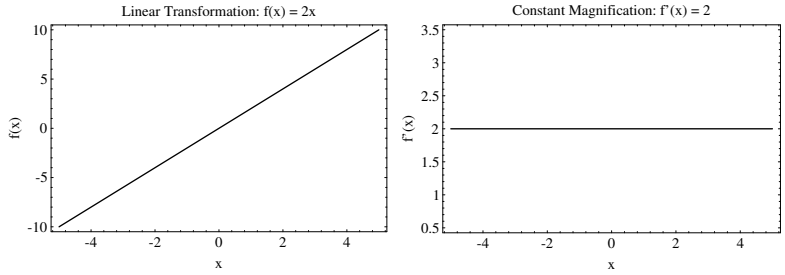

Figure 1: Linear Transformation and Constant Magnification Functions

- The user is forced to create a mapping between disjoint levels of resolution in the image. While it can be argued that the user will already be somewhat familiar with this task, it can also be argued that very rarely in the real-world does our perception of objects follow this type of behaviour: when viewing a physical object, our levels of perceived detail tend to follow a smooth continuum, rather than jump in and out to different levels of resolution.

  We would like to implement a system which more naturally mimics the continuous levels of detail offered in real world systems. The user should be able to gradually draw closer to areas of interest (thus increasing the resolution) and move away from areas of disinterest (decreasing the resolution).

- If the magnified representation of some area of the screen is to appear in some other part of the screen (i.e. a separate window), then the user will be forced to make abrupt transitions on *two* cognitive/perceptual levels to tie the representations together. The first transition is spatial: the eyes (and possibly input devices) will have to move from the normal resolution version of the image to the higher resolution version. The second transition is that of resolution, the user will be forced to create a mapping between the displayed levels of resolution.

- If we attempt to use *in situ* linear magnification techniques, where the magnified image sits on top of the normal resolution image, then we introduce an additional problem, that of *occlusion*. Because of the linear nature of this magnification, the magnified representation of an image must necessarily be larger than the non-magnified representation, the result of this is that the magnified image will block neighboring areas of the non-magnified image, so that the user is no longer able to see the entire image (at any resolution) without moving the magnifying area to see what is underneath it.

2.2 Non-Linear Transformations

In order to overcome some of the problems with linear magnification, several different *non-linear* (or *distortion-based*) magnifications have been developed. These systems all share the properties of enhanced local resolution with some preservation of global context.

2.2.1 Fisheye Zoom

The *fisheye lens* is one mechanism for dealing with the problems incurred by linear magnification. In this schema, the user is provided with a view which is distorted in a manner similar to that produced by a very wide-angle camera lens. Furnas popularized this idea in 1986 [5], and it has since been used in other systems such as Sarkar and Brown [17]. Several different mechanisms can be used to implement the fisheye lens effect. It is not in general necessary to construct an explicit lens model, rather the effect can be achieved through simple geometric transformations. A thorough discussion of one mechanism for implementing fisheye transformations can be found in works by Sarkar and Brown [16] [17].

2.2.2 Hyperbolic

Hyperbolic geometries provide a natural means for producing local resolution enhancement and global context. This approach allows infinite Euclidean space to be mapped into a finite disk in a continuous, non-occluding manner so that the space is "bigger" near the center of the disk, and "smaller" near the periphery. Examples of work using such geometries can be found in [11] and [13].

2.2.3 General Non-Linear

The fisheye lens and hyperbolic geometry are special cases of a class of transformations called *non-linear* or *distortion based* transformations. A general property of these non-linear transformations is localized enhancement of detail while still preserving some of the global context for the viewer. There have been a number of papers describing different approaches to this, a survey of many of these can be found in [12]. The remainder of this subsection will describe some approaches and techniques for non-linear transformations.

- **One Dimensional**

  A desirable property for a 1D base function for non-linear transformations is that it should have a smoothly varying slope over some well defined domain, such that some segment(s) of the curve will have slope > 1 (areas of magnification) and some other segment(s) of the curve

have slope < 1 (areas of *demagnification* or *minification*). In addition, the function chosen should be such that a parameter exists for increasing the slope of the function in some areas, and will also cause a corresponding decrease of the slope in other areas. These requirements on the function reflect the common sense notion that "you can't get something for nothing". Every extra level of magnification must incur a corresponding decrease in magnification at some other location if the range boundaries are not to exceed the domain boundaries (which is a requirement for non-occluding magnification).

Several different choices exist for a 1D base transformation function. Furnas [5] and Sarkar and Brown [17] use the function $G(x) = \frac{(d+1)x}{dx+1}$ to perform this transformation. $G(x)$ is well behaved and has desirable non-linear characteristics over the domain $[0, 1]$, so that $G(x) : [0, 1] \rightarrow [0, 1]$. Although this function is by itself relatively inexpensive computationally, all coordinates to be transformed by this function must first be normalized or constrained to the $[0, 1]$ domain.

Alternatively, it is possible to use a function which is well behaved over an infinite domain $h(x) : \mathcal{R} \rightarrow (-C, C)$. Such a function maps points at infinity to some fixed point. There are many possible functions which can be used to accomplish this, however the most straightforward approach is to use the hyperbolic tangent function $h(x) = tanh(x)$. To this function we can add a parameter β which controls the degree of magnification, so that $h(x, \beta) = \tanh(x\beta)$. $h(x)$ is well behaved across all domains, and there is no need to normalize the domain coordinates to some specific range before applying the magnification. Note also that several functions exist which can produce a similar effect to that of $\tanh(x)$. In particular a modified version of the logistic function ($h(x, \beta) = \frac{2.0}{(1.0+e^{-2\beta x})} - 1.0$) is computationally less expensive on many machines. It will also be shown in section 6.1 that piecewise linear functions can be used to provide a reasonable approximation. Figure 2 shows a comparison of 1D functions for fisheye, tanh and logistic transformation functions at different levels of magnification. Figure 3 shows the tanh transformation function and its associated magnification function.

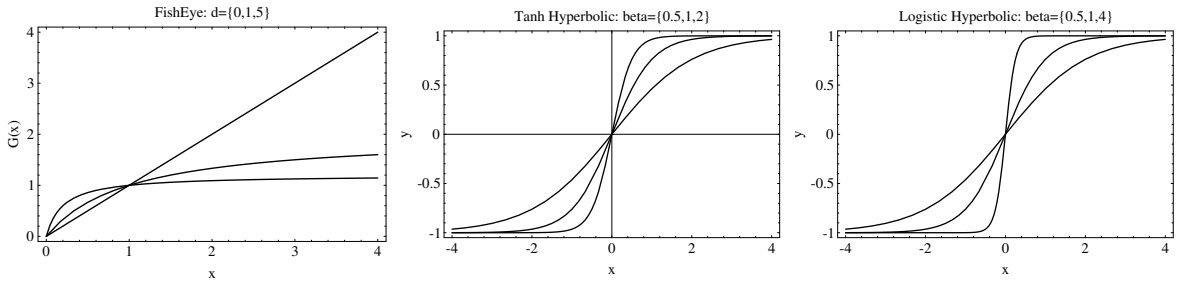

Figure 2: Fisheye, Tanh, and Modified Logistic Basis Functions

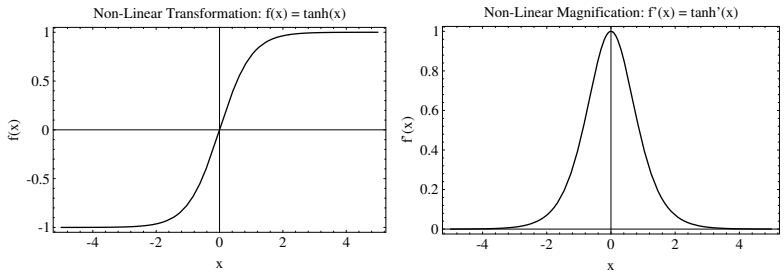

Figure 3: Tanh Transformation and Magnification Functions

What is still required is a mechanism for moving the center of magnification across the domain. This is easily achieved: if we want x_0 to be the center of magnification, we simply replace $h(x)$ with $(h(x - x_0) + x_0)$, and the center of maximal magnification will automatically translate to the desired location.

As will be shown in the following section, a 1D function can be used as the base function for transformations in two or more dimensions. It can also be used for magnification in one dimension only; Keahey and Marley [9] have shown that using such a "magnification bar" can be superior to scrolling for viewing highly structured text documents.

- **Two Dimensional**

  By applying these simple 1D transformation functions to two dimensional coordinate spaces in various ways, many different effects can be produced. Figure 4 shows just a few of the possible effects of applying such transformation to a regular grid of two dimensional points (these are described below). Many other transformations are possible and have been implemented.

  Radial (Fisheye):

  The fisheye lens effect can be produced by transforming each point in the domain as follows:

  1. Given a center point of magnification H and a point to transform P, let $\hat{P} = P - H$
  2. Find the radius component of the polar coordinates of $\hat{P}$ so that $r = \sqrt{\hat{P}_x^2 + \hat{P}_y^2}$
  3. The new coordinates are then $H + \frac{h(r)}{r}\hat{P}$

  This transformation has the nice property that it preserves angles relative to the center of the magnification image. Additionally, the fisheye effect is familiar to most users, and provides a ready analogy for the user to relate to. One potential limitation is that there is only a single magnification parameter, which controls degree of magnification in all directions.

Orthogonal:

This transformation is the result of applying the 1D transformation to the x and y coordinates of a point separately, thus making the magnification in one dimension *orthogonal* to magnification in other dimensions. This transformation has the advantage of preserving horizontal and vertical lines within the domain, as well as allowing for independent control of the magnification parameters for the x and y axes. It has the disadvantage however, of not preserving angles relative to the center of magnification.

Bi-Radial:

The bi-radial transformation is a combination of the radial and orthogonal transformations. The transformation is achieved by making the *direction* of transformation the same as in the radial case, however the *magnitude* is weighted by the x and y component as in the orthogonal transformation. This transformation offers the advantages of preserving angles relative to the center of magnification, as well as providing some degree of independence for x and y magnification parameters.

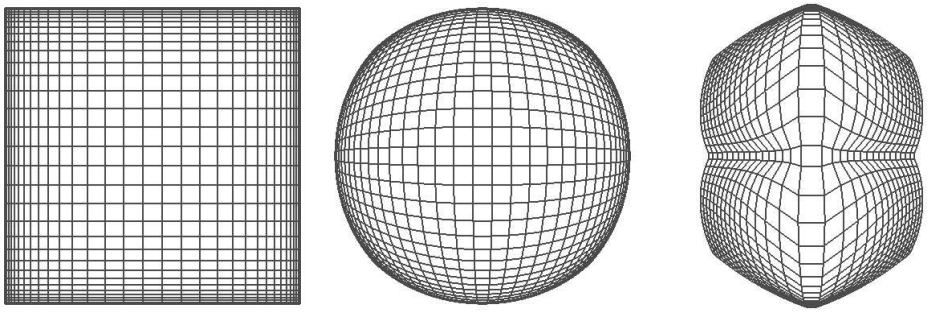

Figure 4: Orthogonal, Radial and Bi-radial Transformations

2.2.4 Combined Linear/Non-Linear

The motivation for linear transformations is that they produce distortion-free zooming (when the magnification factor is the same in both dimensions). For example, a user viewing textual data would prefer to see the letters at the area of maximal magnification without the distortions presented by non-linear transformations. This is relevant with a technique developed by the author where arbitrary images (often containing text) are texture-mapped to a grid which is transformed via the techniques presented in this paper. As the grid points are transformed, the image is transformed along with them.

We can combine the advantages of linear and non-linear transformations by linearly transforming all points within a certain area A, and all points not falling within A are interpolated into the area surrounding the linearly magnified area of A and subsequently transformed with a suitable non-linear transformation. Note that the maximum degree of magnification that can be used in the linearly magnified area is constrained by: 1) the size of the domain to be magnified (inversely

proportional), and 2) the size of the target domain (directly proportional). If we want the entire image to fit in some bounded domain, then we can only linearly magnify an area so far before presenting problems with expansion beyond the boundaries of the target domain.

Figure 11 shows what a combined linear/non-linear transformation might look like on a regular grid of points, and Figure 5 shows two examples of this type of transformation. In the left image, a round area is used for the linear magnification, with the surrounding points being treated by a radial non-linear magnification. In the right image, a rectangular area is used for the linear magnification and the surrounding points are transformed by an orthogonal non-linear magnification.

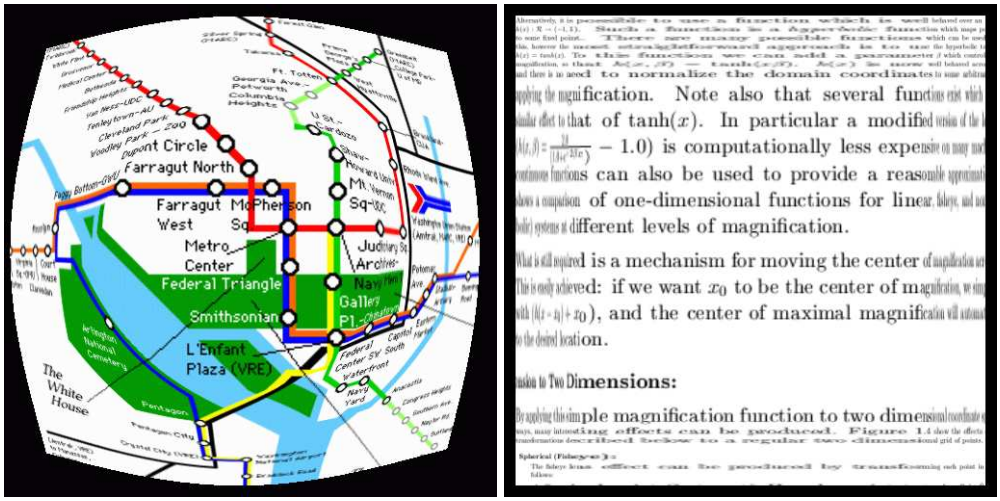

Figure 5: Combining Linear and Non-Linear Transformations

2.2.5 Non-Geometric (Semantic)

This type of transformation typically involves a semantic level change in the representation of objects based on their level of magnification. Very often this is used in conjunction with some type of non-linear transformation. A full discussion of these approaches is beyond the scope of this paper, however the techniques that we present here are relevant to many of them. The interested reader is referred to [2], [1], [4] and [7].

2.3 Continuous/Discrete Domain Characteristics

For tasks such as graph visualization, the domain consists of discrete objects in a continuous coordinate space, and (in general) adjacencies between nodes can be maintained through rendering of edges directly between then, regardless of the type or level of distortion introduced by the transformation. For this reason, graph visualization tasks are very amenable to non-linear transformation techniques, and there are many graph visualization systems which take advantage of this [14], [10], [17] and [5].

For discrete domains (such as the texture mapping example just presented), adjacency requirements become more of a concern. This is particularly the case when linear/non-linear combinations or constrained boundaries are introduced. In these cases, boundaries between areas of different transformations must be carefully thought out to preserve adjacency information. The basic task is to ensure that all mappings between areas generate boundary conditions which are consistent with the other transformation functions that are used. Referring to figure 6, we can say that the boundary conditions at the perimeter of A should be the same for both $f(A)$ and $g(\sim A)$.

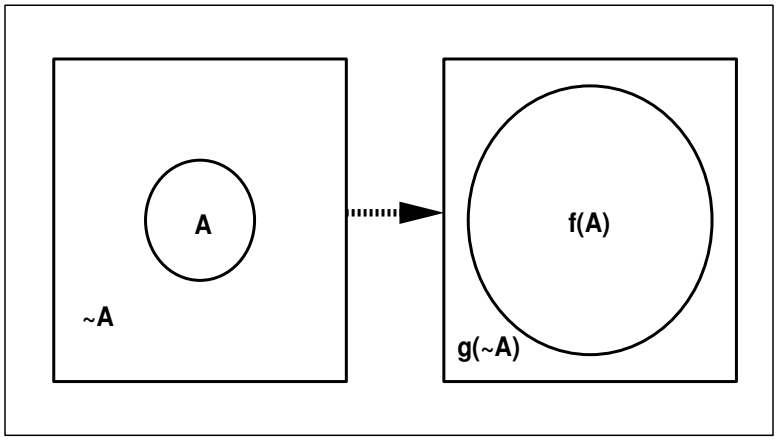

Figure 6: Boundary Conditions

3 Constraining Transformation Domains

Often it is the case that we do not want the transformation to apply to the entire source domain, but would rather perform non-occluding magnification on some sub-area of the domain, constraining the transformed points from that sub-area to the same sub-area (to avoid overlap). This allows for a localized, non-occluding magnification which can be moved over the source domain. This localization offers the benefits that the global context now remains more static, and the boundaries of that context remain fixed even as the center of the magnification is changed. Figure 7 shows examples of applying these constrained domains to the texture mapping example described above, compare these results with Figure 5.

The mechanism for performing these domain constrained transformation is similar to that used for the linear/non-linear combined transformations described in section 2.2.4, except that we perform the non-linear transformation *inside* the sub-area of the domain, and all points outside the domain remain untransformed. Note that it is often desirable to map the points inside the sub-area to some regular domain (e.g. $[-1,1] \times [-1,1]$) before performing the actual transformation on them, and then mapping the transformed points back into the original sub-area. If we choose a transformation function with a fixed range (such as $\mathcal{R} \rightarrow (-1,1)$ these mappings are more easily facilitated.

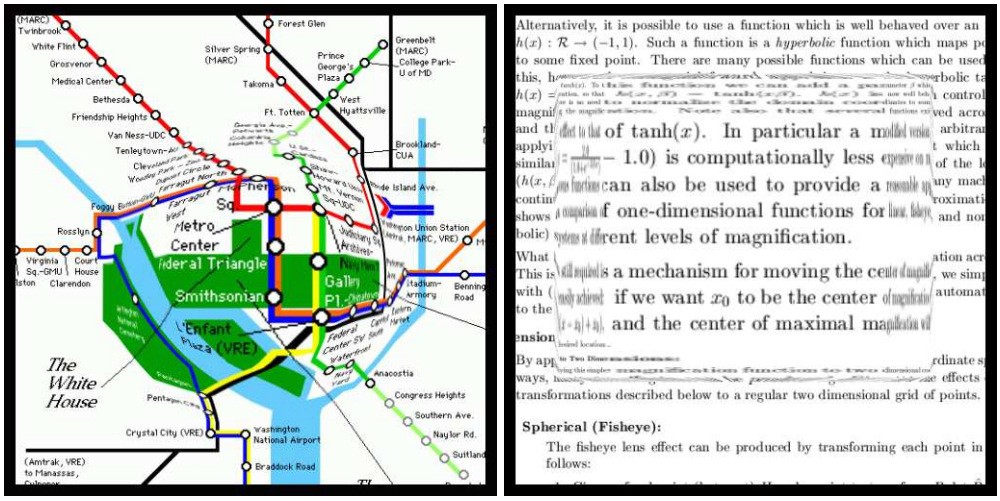

Figure 7: Constraining Transformation Domains

4 Compound Transformations

Work on static transformations with multiple areas of magnification began with [8]. Other authors [5] [16] [17] have discussed the possibility of combining multiple centers of magnification within a single domain. This section will consider methods for combining transformations from multiple individual transformations. These techniques fall into three general categories, each with their own distinct questions and characteristics: 1) restricting the transformations so they do not overlap, 2) globally combining the transformations, and 3) applying the transformations in succession. The images in Figure 8 are the result of applying instances of these three approaches to a regular two dimensional grid.

Maximal Ray Clipping:

In this schema, each point in the domain is transformed only by the transformation whose center of magnification is closest to that point. After the transformation has been performed, the point is clipped back along the ray between itself and the transformation center (if any other centers of magnifications are closer to the transformed point) until the clipped point is again closer to the original transformation center than to any other transformation center.

This approach produces a partitioning of the domain space similar to Voronoi diagram partitioning methods from computational geometry [15]. The effect of this technique is a partitioning of the space in which each transformation has its own "domain of influence".

Weighted Averaging:

This is achieved by independently applying each transformation to a point in the original domain, and then making the final transformed point the weighted average of the independent transformations. The weights are inversely proportional to the distance between the center of magnification and the point in the original domain.

Composition:

Here we apply each transformation in sequence to the points in the domain. This raises issues of what order the transformations should be applied in, since the magnifications will in general be non-commutative. The system used to produce the images below allowed for this through an interface to a stack of transformations, and all transformations were applied to the domain in order from the bottom to the top of the stack. The effect of this is that of a stack of lenses layered on top of the display, each of which can be controlled independently.

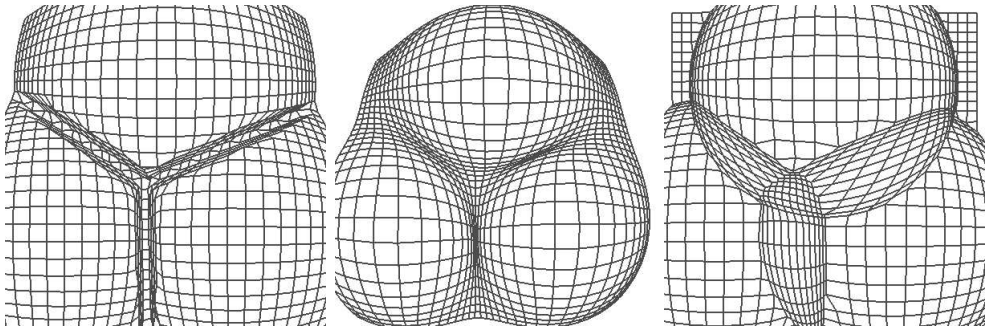

Figure 8: Clipped, Weighted Average, and Composition Transformations

5 Filtering Transformations

Independent of the complexity of the transformations employed, it is often useful to have a single mechanism for determining the degree to which the warping transformations are to take effect. This can allow the user to smoothly zoom between the warped and unwarped representations of the space being displayed. Such functionality can be provided through a simple filter applied to the transformed and untransformed points. This filter allows a variable weighting of the points. Let s be the weight attached to the source point, and d be the weight attached to the transformed point. We make the conditions that $0 \leq s \leq 1$ and $d = 1 - s$. Figure 9 shows an example of changing the filter parameters to alter the degree of the effect of several transformations simultaneously.

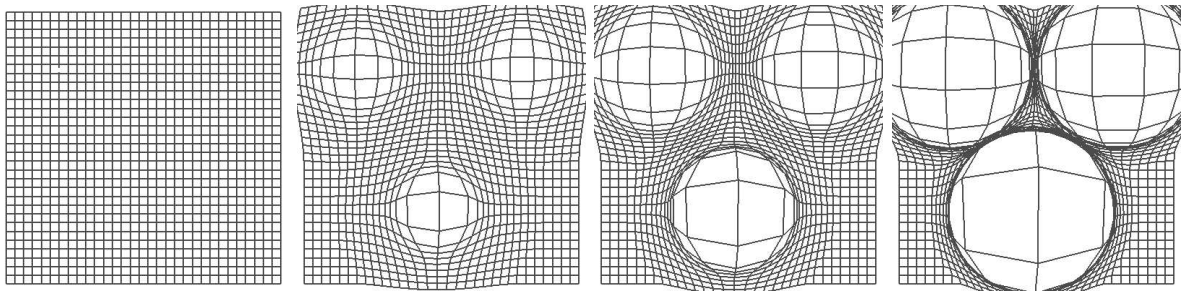

Figure 9: Filtered Transformations for $s = 1.0, 0.65, 0.35, 0.0$

6 Piecewise Transformations

In addition to the continuous transformations described above, it is also possible to approximate these transformations using piecewise linear functions. Such functions offer the potential for performance gains, and also allow for a more general class of transformations.

6.1 Simple Piecewise Transformations

This class of transformations involves piecewise linear approximations of the single-variable transformation basis functions which are used to drive the higher-order (2D and 3D) transformations (as described in section 2.2.3). As an example of this, a piecewise approximation of $\tanh(x)$ can be used to replace that computationally expensive non-linear function call with a simple table lookup and a linear interpolation. Since we can easily parameterize the resolution of the piecewise approximation, it is possible to produce a step function that is arbitrarily close to the continuous one, at a reduced computational cost. Table 1 shows some timing results [3] for the simple radial transformation using different 1D basis functions which are shown in Figure 10.

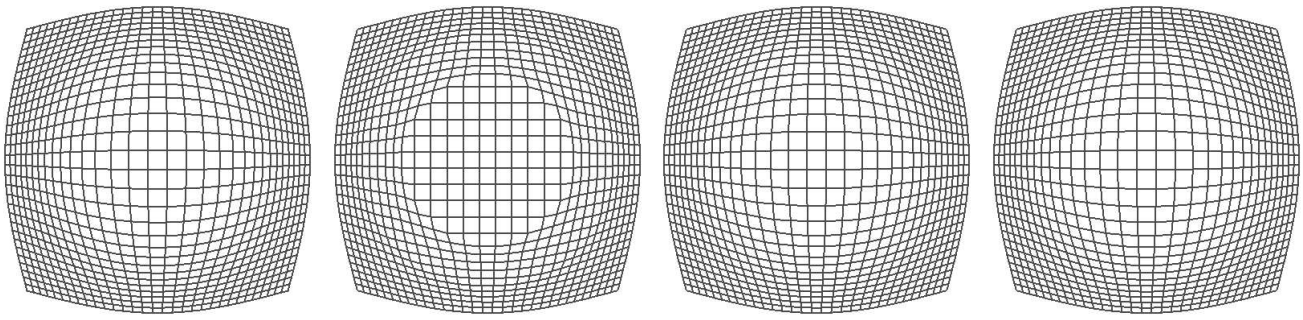

Figure 10: Comparison of 1D Bases for Radial: tanh, Piecewise ($n = 8, 16, 32$)

Note that Table 1 shows the time for the piecewise approximations to be slightly slower than the fisheye transformation time, and that there is no time increase for the piecewise function as the number of components in the approximation is increased. However, the expressive power of the fisheye basis transformation is limited to altering the degree of curvature of the function based on a single parameter. With piecewise functions on the other hand, it is possible to transform points with 1D transformations of arbitrary complexity at no extra computational cost.

We can see a simple example of the additional expressiveness of the piecewise transformation, by considering the methods used for producing the linear/non-linear combined transformations

[3] This timing results were obtained on a 150MHZ MIPS R4400 processor using the sum of *stime* and *utime* as reported by the unix times() command. Times shown are average times (in seconds) for computing a transformation on a single point (based on applying the transformation to a grid of points multiple times). Normalized times reflect the difference between the transformation time for the distortion function and a "Null" transformation which returns the original point, thus eliminating overhead common to all transformations (such as calling overhead).

Function	Time	Normalized
Null	2.15e-06	0.0
tanh	7.23e-06	5.08e-06
logistic	6.64e-06	4.49e-06
fisheye	5.47e-06	3.32e-06
Piecewise 8 Steps	6.05e-06	3.90e-06
Piecewise 16 Steps	6.05e-06	3.90e-06
Piecewise 32 Steps	6.05e-06	3.90e-06

Table 1: Timing Results for Radial Transformation With Different Bases

discussed in section 2.2.4. Using a continuous warping function such as *tanh* or the fisheye transformation, it is necessary to treat points inside the area of linear magnification as a special case. However piecewise linear transformations can automatically provide the desired area of flat magnification. Figure 11 shows a comparison of the results of these transformations (the piecewise version was obtained by using the piecewise radial transformation with 6 segments). Table 2 shows some timing results for the different methods (the final entries in this figure and table will be described in the following section). Note that the times for the piecewise function are identical to the times for the simpler step radial function in Table 1. These data clearly show a performance potential for the piecewise transformation.

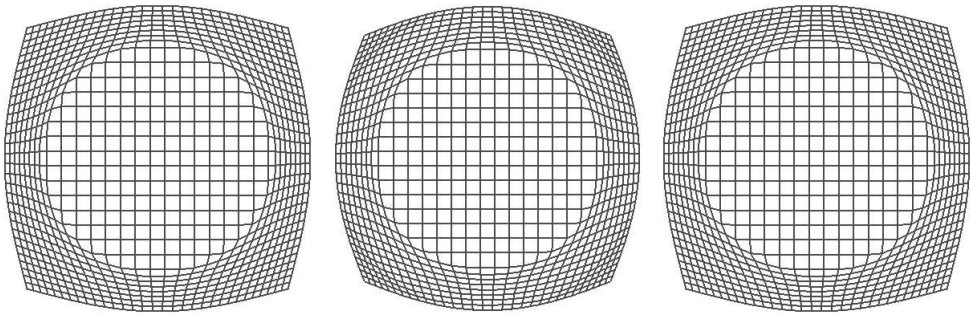

Figure 11: Flat/Radial Transformation for Continuous, Simple and Complex Piecewise Bases

Function	Time	Normalized
tanh	1.30e-05	1.09e-05
fisheye	1.19e-05	9.75e-06
Simple Piecewise	6.05e-06	3.90e-06
Complex Piecewise	7.42e-06	5.27e-06

Table 2: Timing Results for Flat/Radial Transformation With Different Bases

In addition to the performance gains offered by such functions however, piecewise functions generalize much more readily to arbitrary shapes. In general we can say that a suitable piecewise transformation function is one composed of linear segments having zero-order parametric continuity C^0 (i.e. the endpoints of adjacent segments meet). The degree of magnification provided by the function is directly proportional to the slope of the segment at that point. Segments with slope

greater than 1 will magnify, and segments with slope less than 1 will minify.

There are several ways in which such functions can be constructed. The simple piecewise functions used in this paper were all generated through automatic routines which sample a continuous function at a parameterized resolution to obtain the resulting piecewise function.

It is also possible to provide an interface which allows an user to drag control points on the piecewise transformation function and thus design customized functions with multiple areas of magnification. Not all functions constructed in this manner will be useful, the loose restriction of C^0 continuity will allow the user to construct wildly distorting transformations with sharp transitions rather than smooth curves. This problem can be alleviated by using spline functions with higher degrees of continuity [3] to construct a "transformation curve", and then sampling the spline function to obtain an approximating piecewise linear function.

Piecewise transformations can also be constructed through manipulation of the values of the magnification function (which is the derivative of the transformation function). By placing a grid over some domain and creating suitable magnification values (perhaps corresponding to a degree of interest) for each cell in grid, we can integrate over the magnification values to obtain the corresponding transformation function. This allows the user to construct a transformation function without having to be aware of the slope/magnification relation, instead the user is able to construct such a function by simply expressing a degree of interest in certain areas of the domain. Since we are integrating the magnification values to obtain the transformation function, we automatically obtain C^0 continuity in the resulting transformation function. As with the direct construction of transformation functions described in the previous paragraph, this process is also amenable to the use of spline functions to smooth out the construction of these values.

6.2 Complex Piecewise Transformations

In the previous subsection, we explored methods for using piecewise linear functions to approximate the 1D basis functions described in section 2.2.3. Such piecewise approximations are called *simple* piecewise transformations because we are approximating single variable functions. In this subsection, we will examine a more powerful set of piecewise transformations, where an approximation is made of a complete two dimensional transformation. These approximations are called *complex* piecewise transformations, since we are now approximating multi-variable functions.

The most straightforward method to construct a complex piecewise linear transformation of this type is to sample an existing transformation with a regular grid of two dimensional points to construct a grid of the transformed points. After this grid has been computed, any number of points can be transformed through table lookup and linear interpolation on the x and y coordinates independently. The accuracy of the piecewise approximation is determined by the resolution of the "sampling grid" that is used [4].

[4] In the system developed by the author, grid resolution is parameterized and user-controllable, both for complex and simple piecewise transformations.

Table 2 shows that even for *slightly* complicated transformations such as the Flat/Radial combination (shown in Figure 11), construction of a complex piecewise transformation can result in significantly faster transformation times than for the procedural fisheye technique. As with the simple piecewise transform, the time required for the complex piecewise transformation remains constant with increasing complexity of the function, and increasing grid size (within memory and cache constraints). Note however that the simple piecewise approach is the fastest of the three, requires much less memory than the complex, and would be a better choice for this particular transformation.

When we moved from continuous basis transformation functions to their simple piecewise transformation counterparts, we found that a new level of expressiveness became possible. Similarly, we can express an even richer set of transformations with a single complex piecewise transformation function than with a continuous 2D transformation.

There are several ways in which this potential increase in expressiveness can be realized. One method that offers large performance benefits is the ability to combine multiple transformations (as described in section 4) and express them in terms of a single complex piecewise transformation. This can be achieved in a manner similar to that described above for construction from a single existing transformation, the difference being that the entire set of transformations is applied to the sampling grid before it is used to construct the single complex piecewise transformation grid. The timing results shown in Table 3 [5] indicate better than a magnitude of order increase in performance when comparing the procedural (fisheye) and complex piecewise approaches to producing the transformation shown in Figure 12.

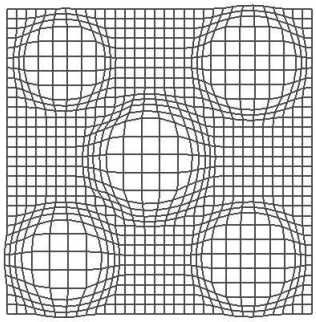

Figure 12: Multiple Transformations used for Table 3

Function	Time	Normalized
Null	4.59e-06	0.0
Fisheye	7.10e-05	6.64e-05
Complex Piecewise	1.02e-05	5.61e-06

Table 3: Timing Results for Multiple and Complex Transformations

Because of the overhead involved in construction of complex piecewise transformations, they are

[5] The timing for the Null transformation is different in this case than for previous timings. The difference is reflective of the fact that we now have the additional overhead of passing points through a *list* (possibly of length 1) of transformations rather than through a single transformation.

best suited to tasks which are dynamic in time and placement rather than in shape. Once computed, these complex transforms can be translated over a domain (or the domain "underneath" the transform can change through time) efficiently without recomputing the transformation grid. Whenever the *shape* of the transformation changes however, it becomes necessary to recompute the transformation grid, thus simple piecewise or continuous functions might be better suited to those applications where the shape of the transformation changes frequently.

The same techniques for constructing simple piecewise transformations through direct manipulation of either the transformation or magnification values can also be applied to complex transformations. However additional issues arise in increased complexity of the user interaction task, as well as in methods for effectively converting magnification values to the corresponding transformation function.

7 Conclusions

In this paper we have shown several techniques for constructing and using non-linear magnification. We have shown that it is possible to construct sophisticated transformations by building up from simpler ones. A hierarchy of transformations emerges from these constructions which allows for categorization of new classes of transformations. Some of the construction tools that we have introduced include: combination with linear magnifications, constrained transformation domains, combining multiple transformations, and enhanced control of the overall degree to which a transformation(s) should take effect.

In addition, we have shown how piecewise linear transformations can be used to approximate continuous non-linear transformations of one or two dimensions. Timing data clearly indicate a potential for increased efficiency with these piecewise transformations. In addition, a greater degree of expressiveness becomes possible with the piecewise functions, which provides greater flexibility for the designer of more sophiticated magnification systems.

8 Further Work

General methods for enforcing the consistency of domain boundary conditions in aggregate transformations should be addressed. These problems need to be addressed in the areas of domain constraints and combinations of transformations.

More work is currently being done on effective techniques for constructing the piecewise linear transformations. We are investigating interactive methods for construction of these transforms which will be more natural for an user to deal with. Also, several issues arise in the conversion between the transformation and magnification grids for the complex piecewise functions. Different conversion techniques exist, each giving different results. A significant area of work centers on

the question of how to best utilize the additional expressiveness possible with these piecewise transformations.

References

[1] Benjamin B. Bederson and James D. Hollan. Pad++: A zoomable graphical interface. In *CHI '94 Proceedings*, 1994.

[2] Benjamin B. Bederson and James D. Hollan. Pad++: A zooming graphical interface for exploring alternate interface physics. In *UIST '94 Proceedings*, 1994.

[3] James D. Foley [et al]. *Computer Graphics: Principles and Practice.* Addison-Wesley, 1990.

[4] Ken Fishkin and Maureen C. Stone. Enhanced dynamic queries via movable filters. In *CHI '95 Proceedings*, 1995.

[5] G. W. Furnas. Generalized fisheye views. *Human Factors in Computing Systems, CHI '86*, pages 16–23, April 1986.

[6] George W. Furnas and Benjamin B. Bederson. Space-scale diagrams: Understanding multiscale interfaces. In *Proceedings of CHI '95 Human Factors in Computing Systems*, 1995.

[7] Brian Johnson and Ben Shneiderman. Treemaps: A space-filling approach to the visualization of hierarchical information structures. In *IEEE Visualization '91 Proceedings*, 1991.

[8] N. Kadmon and E. Shlomi. A polyfocal projection for statistical surfaces. *Cartograph. J.*, 15(1):36–41, 1978.

[9] T. Alan Keahey and Julianne Marley. Non-linear in situ magnification techniques: Overview and an experimental study. Term project conducted at Indiana University SLIS, Dec. 1995.

[10] T. Alan Keahey and Reid Rivenburgh. Hyperlink: A program for visualizing traversal of the www. Demonstrated at HyperText '96, 1996.

[11] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI '95 Proceedings*, 1995.

[12] Y.K. Leung and M.D. Apperly. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

[13] Tamara Munzner and Paul Burchard. Visualizing the structure of the world wide web in 3d hyperbolic space. In *VRML '95 Proceedings*, 1995.

[14] Emanuel G. Noik. Exploring large hyperdocuments: Fisheye views of nested networks. In *Hypertext '93: ACM Conference on Hypertext and Hyeprmedia*, 1993.

[15] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, Berlin, Heidelberg, Tokyo, 1985.

[16] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. Technical Report 84, Systems Research Center, 1992.

[17] Manojit Sarkar and Marc H. Brown. Graphical fisheye views. *Communications of the ACM*, 37(12):73–84, 1994.