NONLINEAR MAGNIFICATION

T. Alan Keahey

Submitted to the faculty of the Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in the Department of Computer Science

Indiana University

December 1997

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

Edward L. Robertson, Ph.D.

Doctoral Committee

Randall Bramley

Andrew Hanson

Dirk Van Gucht

December 2, 1997

Andrew Dillon

for my wife

and for my parents

the original Doctors Keahey

and for the memory of my grandfathers

Nels Gustaf Jerde and Thomas Elbert Keahey

Acknowledgements

There are a number of people who have contributed immensely to my success and happiness, beginning with my parents who proved by example that it is possible to go through this entire process and still maintain a healthy perspective on life. I'm lucky to have had the support of my three sisters Deanna, Debbie and Laine, as well as the constant support of my grandparents. I hope that this thesis in some way justifies my grandmother Ida's confidence that I would find my niche in life.

Ed Robertson has done a great job of encouraging open exploration of new ideas while still remaining focussed on the key issues, his guidance has helped me to see things in a larger perspective. These last years of thesis research have not only been intellectually stimulating, but also personally enjoyable. Thanks to my dissertation committee members for their insightful comments and constructive suggestions before, during and after the writing of this thesis.

Craig Stewart was a great boss during my early days of graduate school. David

Leake was the first faculty member here to say "hello" to me in the hallway. Dirk Van Gucht provided early encouragement and kept the department at arm's length while I looked for a research area to call my own. Andy Hanson and Pete Shirley are largely responsible for awakening my interest in computer graphics. Special thanks to Randy Bramley for venturing into the lab from time to time to share his insights and stories about Computer Science, the department, academia, and life in general. The facilities staff here does a world-class job, Rob Henderson and Bruce Shei have been especially helpful. Pam Larson also smoothed my progress administratively. This thesis was supported by U.S. Department of Education Award P200A502367, and partially supported by NSF Grant CDA-9601632.

Some of the cool people that I co-inhabited the graphics lab with are: Tom Loos, Eric Wernert, Kurt Zimmerman, Gladimir Baranoski, Younghee Lee, Chun-Perng Cheah, Anuradha Padte, Yun Bai, and Thomas Stuckey. Eric Wernert is a great guy and fun to teach with. Tom Loos has a laugh that still bounces through the corridors of Lindley Hall, conversations with him are never dull. Some other students with whom I have interacted are: John Rehling, Beata Winnicka, Chuck Shepherd, Bill Dueber, Gennis Emerson, Kaushik Mody, Munish Gandhi, and Julianne Marley.

Lastly and most importantly, my wife Kate has enriched my life so deeply that the peaks and troughs of the graduate school experience reduce to ripples on the surface. None of this would have been possible without her support and understanding.

Abstract

T. Alan Keahey

Nonlinear Magnification

In this dissertation we will explore nonlinear magnification as a generalization of the familiar concepts of "fisheye" or "distortion-oriented" views, which produce in-place magnification while preserving visual context. We will formalize this generalization in order to provide a rigorous basis on which to understand and develop spatial nonlinear magnification systems. At the core of our theoretical basis for nonlinear magnification lies the distinction between transformation functions which directly transform the spatial coordinates, and magnification functions which reflect the degree of magnification that is implicit within the transformation. This relationship was previously established for 1D functions, and we will now extend this to functions of higher dimensions. Doing so allows us to quantify the effects of transformations from a wide variety of nonlinear magnification systems, rather than merely observing them as visual or implementational phenomena.

We will examine two new systems for producing nonlinear magnification. The first method uses a RISC-type approach to construct a transformation pipeline composed of sequences of simple and modular 2D transformations. This transformation pipeline is able to produce complex transformations, and is also very efficient computationally. The expressiveness and efficiency of this system is greatly facilitated by the use of piecewise linear functions which reduce complex transformations to simple table look-up operations.

The second method introduces the nonlinear magnification field as a low-level representation based on our formalism of the relation between transformation and

magnification functions. We will see how the implicit magnification field of a transformation can be computed, and will also provide an iterative method for constructing a transformation from a specified magnification field. The scalar magnification field representation is particularly amenable to user and program manipulation. Because there are no restrictions of explicit foci or multi-dimensional dependencies, direct specification of the desired magnification values is now possible.

We will also develop a general framework for describing the levels on which nonlinear magnification can be applied, and examine the issue of how to effectively synchronize detail rendering functions to take advantage of the extra space produced by nonlinear magnification transformations.

Contents

List of Tables

List of Figures

1

Introduction

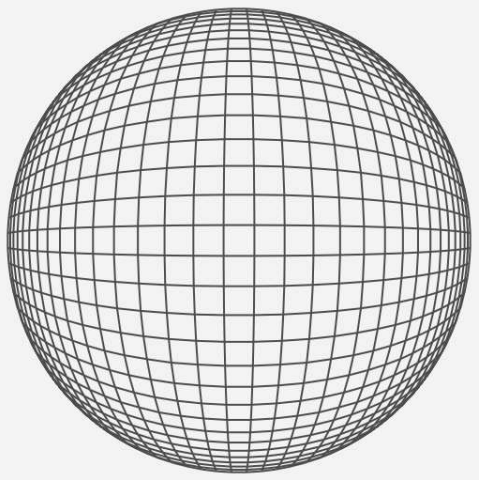


Figure 1.1: Nonlinear Magnification

We begin our exploration of nonlinear magnification with a simple visual example, as shown above. This example shows the effect of transforming a regular grid, and represents one of the simplest instances of nonlinear magnification, illustrating how space can be transformed to expand and compress (magnify and demagnify) regions. Nonlinear magnification offers many advantages over the traditional concept

of magnification. The major distinguishing characteristics of nonlinear magnification are:

- *enhanced resolution of areas of interest*
- *preservation of global context*
- *in-situ magnification* (detail is shown within context, not in a separate window)
- *non-occlusion* (magnification of one area does not block out surrounding areas)

Traditional magnification techniques are unable to simultaneously satisfy all of these characteristics, but nonlinear magnification is able to incorporate all of them. An example of the application of nonlinear magnification might be a floating magnification window on a computer screen which tracks the mouse cursor and magnifies the region of the screen under the cursor, in a manner similar to that shown in Figure 1.2.

There has been a great deal of research on techniques related to nonlinear magnification, much of which is described in Chapter 5. Much of the existing work is not well-unified, but is instead a scattering of approaches to achieving similar visual results. As evidence for the non-unified nature of research in the field, consider some of the different terms which are used in the literature to describe the techniques: *distortion oriented presentation*, *fisheye views*, *focus+context*, *multi-viewpoint perspective display*, *zooming graphical interface*, *polyfocal projection*, *hyperbolic space*,

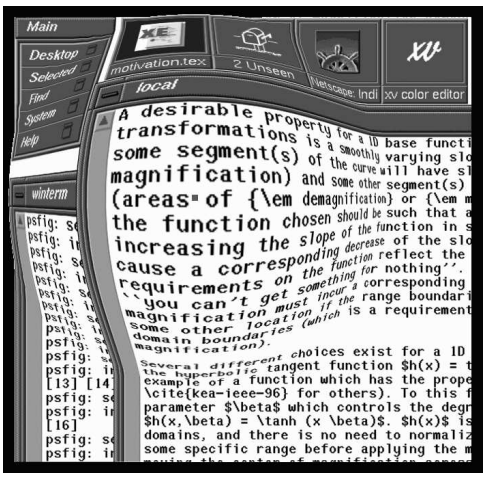


Figure 1.2: An Application of Nonlinear Magnification

pliable surfaces, *alternate interface physics* and *stretching the rubber sheet*. A formal abstraction for describing and developing nonlinear magnification systems has been lacking. Much of the current research is described only in terms of visual metaphors or specific implementation details. While such descriptions might be adequate for describing individual systems (particularly at the user level), they do not facilitate comparisons between systems or provide a general theory of nonlinear magnification. Conceptual rigour is needed to promote understanding and development of the core concepts involved in nonlinear magnification.

Towards that end, the term *nonlinear magnification* was introduced in [KR96c, KR96d] to describe the effects common to all of these techniques. The focus of this thesis will be to develop conceptual frameworks and implementations which unify the current notions of nonlinear magnification. Two new paradigms for nonlinear

magnification will be described in this thesis, each of which represents a different approach to satisfying a number of design guidelines for a framework for nonlinear magnification:

- provide and describe the "transformational functionality" of existing approaches to nonlinear magnification
- identify and eliminate the shortcomings of existing approaches to nonlinear magnification
- provide a systematic model on which to base further development and understanding of nonlinear magnification
- be general and flexible enough to allow extension into novel magnification techniques and problem domains
- be efficient enough to allow for the application of nonlinear magnification to real-world problem sizes

Related to the nonlinear magnification framework issues described above, this research will also provide a framework for the different levels on which nonlinear magnification can be applied. Much of the current research involves graph visualization, a task to which nonlinear magnification is well suited; in Chapter 4 we will examine a categorization of other levels at which these techniques can be used, such

as image magnification and data-driven magnification. Within this framework, we will also explore how the magnification tools developed in this thesis can be used to facilitate the use of nonlinear magnification in ways which were not previously possible.

1.1 Conceptual Overview

This section describes the major points of the thesis at a conceptual level. The intent is to provide a context in which to further explore the issues relating to nonlinear magnification. Full details of the points raised here will be provided in following chapters.

1.1.1 Background

The basic concept of "nonlinear magnification" is not a new one, although the term itself was introduced only recently in [KR96d]. As early as 1978 Kadmon and Shlomi [KS78] described a technique called "polyfocal projection" for transforming cartographic maps to enhance resolution at areas of interest while still preserving a view of the global context. Although polyfocal projection was implemented only as a FORTRAN program providing static images for a physical plotting device, in many respects the work they described was well ahead of its time. Their system had a

simple mathematical specification, and was able to produce expressive and complex transformations such as were not seen within the computer graphics and visualization community until around 1993.

As a computer interaction task, nonlinear magnification first appears in George Furnas' report on "fisheye views" of structured text files [Fur81]. This report described ways in which sections of text can be collapsed to hide details, while preserving a view of high level features. Although the report does not describe actual magnification *per se*, it represents an early and relevant work in the field. Closely following this report was Spence and Apperley's "Bifocal Display" [SA82], which described a simple two level display with a central region of high magnification and flanking regions at a lower level of magnification. Here we begin to see the idea of using multiple scale versions of data within a single view, an idea which was developed further 10 years later in Mackinlay, Robertson and Card's "Perspective Wall" [MRC91]. The perspective wall represents the first example of a perspective-based approach to nonlinear magnification; perspective projections of a wall in 3D space with a central proximate region, and two wing segments which angle back from the viewer provide a central magnified area with variably demagnified areas flanking either side. The effect of nonlinearly magnified 2D data is produced by mapping that data onto this 3D surface.

The three years which followed (1992-1994) produced a flurry of new techniques for

producing nonlinear magnification effects. Work on graphical fisheye views [SB92a] and rubber sheets [SSTR93] introduced and extended the transformation-based approach to nonlinear magnification by showing how the 2D coordinates of the underlying space could be transformed directly. The document lens [RM93] was introduced as a perspective-based approach addressing some of the shortcomings of the perspective wall. Work on hyperbolic spaces [PG92] revealed an efficient and mathematically rigorous method for producing nonlinear magnification. A controlled user study comparing fisheye and full-zoom methods [SZaD+93] provided support for the suitability of using nonlinear magnification for graph interaction tasks. Noik's work on fisheye views of nested networks [Noi93a, Noi93b, Noi94a, Noi94b] explored the benefits of applying nonlinear magnification to graph layout and topology, and Pad++ [BH94a] expanded the frontiers of document interaction from within a nonlinearly magnified interface.

In 1994 Leung and Apperley provided the first major attempt to unify these widely disparate techniques in "A Review and Taxonomy of Distortion-Oriented Presentation Techniques" [LA94]. This work makes major breakthroughs in explaining the underlying causes and effects of nonlinear magnification. Perhaps the most significant contribution of this work comes from the establishment of the relationship between "transformation" and "magnification". Transformation functions are used to transform points within the 2D space, and the magnification function is the derivative of

the transformation function, representing the "amount" of transformation. This distinction between transformation and magnification plays a central role in the research described in this thesis.

Since 1994, a number of new techniques have been introduced in the literature. Hyperbolic techniques have been extended to 3D space [MB95], and the perspective-based approach has been been extended in new ways [CCF95a, MK97]. Of particular note is Noik's Ph.D. thesis [Noi96], which tied together much of his earlier work to provide a thorough investigation and development of the systems of the time, particularly as they relate to graph visualization tasks. Figure 1.3 shows the growth of publications relating to nonlinear magnification over the previous 20 years, the rapid growth shows this to be a relatively new and expanding research area.

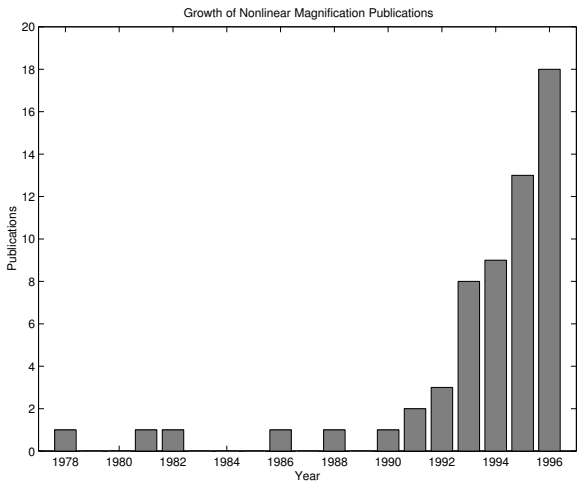


Figure 1.3: Growth of Nonlinear Magnification Related Publications

It should be evident from this brief discussion that the concept of nonlinear magnification touches on many widely different approaches, from simple deletion of lines of text, to continuous spatial transformations, to transformations operating on the topology of hierarchical graphs. Most of the research described in this dissertation will focus on nonlinear magnification of continuous spatial domains. Many of the other transformation domains (such as graph transformations) can be derived from transformations to the underlying spatial coordinates [Noi94b], and thus spatial transformations represent a low level domain of general applicability to focus our attention on. Later sections will also discuss some of the non-spatial aspects of nonlinear magnification.

Now that the preceding discussion has provided a context, we are ready to begin exploration of the ideas developed during this thesis research. The following two sections will briefly describe two new systems for nonlinear magnification, followed by a brief overview of some of the ways in which nonlinear magnification can be applied.

1.1.2 Transformation-Based Magnification

The first approach to developing a nonlinear magnification system involves reduction of complex magnification effects to a "transformation pipeline" composed of simple and modular transformational components. This transformation pipeline

represents a RISC-type approach to nonlinear magnification, in which complex magnification effects can be constructed from sequences of simple transformation functions. Each individual function is typically no more complex than a simple linear interpolation, but by combining these functions in various ways a wide range of effects can be produced. Since each function transforms coordinates directly, this represents another example of the *transformation-based* approach to nonlinear magnification; a more expressive and efficient successor to earlier transformation-based systems such as [KS78, SB92a, SSTR93].

A significant contribution of the transformation-based pipeline approach described here is the ability to increase efficiency and expressiveness through the use of piecewise linear functions. These piecewise linear techniques allow entire sections of the transformation pipeline to be collapsed into table-lookup operations, with a corresponding increase in efficiency. In addition, they facilitate direct manipulation by the user and/or program, thus allowing for a far more expressive class of transformations, and should also prove to be amenable to hardware acceleration.

1.1.3 Magnification-Based Magnification

The second approach to developing a nonlinear magnification system involves the expression of nonlinear magnification as a scalar field of magnification values. These *nonlinear magnification fields* represent perhaps the most elemental approach possible

for dealing with nonlinear magnification effects.

The first level on which these nonlinear magnification fields can be used is for analyzing and comparing the effects of different nonlinear magnification transformations; for any such transformation we can now compute its *implicit magnification field* to find the actual magnification values that are implicit in the transformation's expansion and compression of regions. These fields can be viewed as magnification landscapes, visual aids which assist viewers in perceiving the effects of complex transformations. Additionally, implicit magnification fields are useful on a programming level, providing a numerical method for determining local magnification levels which facilitates techniques such as level-of-detail rendering of objects within the transformed space.

The second level on which these nonlinear magnification fields can be used is as a means for *directly specifying* magnification values within a domain. We will examine an iterative method which constructs suitable transformations from a specified magnification field. By isolating the magnification specification task from the mechanism used to achieve the magnifying transformation in this way, we greatly reduce the complexity of the specification task. Because this technique works directly with the magnification values, rather than relying on the side-effects of transformations or perspective projections, it represents a new *magnification-based* paradigm for producing nonlinear magnification.

We will examine a number of ways in which the ease of expression can be exploited,

both at the user and program level. The level of expressiveness which is possible with nonlinear magnification fields greatly exceeds what is possible with other existing nonlinear magnification systems. Notable examples include *magnification brushing* techniques which allow for subtle magnification effects (particularly useful when applied to image magnification), and *data-driven magnification*, which allows properties of the data being visualized to directly and fluidly define the complex magnification that is best suited to visualizing that same data.

1.1.4 Applications of Nonlinear Magnification

There are a number of levels on which nonlinear magnification can be applied. We will examine three such levels and illustrate the characteristics of each level with specific applications. In brief, the three levels are *image-level*, *render-level* and *data-level*. The primary distinction between these levels is "proximity" of the magnification routines to the data being visualized, although there can be some overlap between levels.

For image-level applications, there is no notion of discrete data objects, rather the magnification operates on pre-rendered data (images). Render-level applications involve transformations of discrete object coordinates to set the location for rendering the object, but do not affect the actual rendering of the objects themselves. Data-level magnifications represent the closest proximity to the data; here the magnification

actually changes the way in which the individual data objects are represented, possibly also tying data-access routines to the level of magnification. Another example of data-level magnifications involves using properties of the data itself to directly define magnification.

An additional issue for applications of nonlinear magnification involves the detail-in-context problem. How can we effectively combine complex magnification transformations with rendering "detail" functions to provide enhanced detail at areas of expansion, and reduced detail at areas of compression? To answer this problem, implicit magnification fields can be used to effectively synchronize the transformation and detail functions. On the implementation side, we will also see how texture mapping techniques can be used to seamlessly integrate different global views of an information space.

1.2 Contributions of the Dissertation

The following items outline the major contributions of this thesis.

- provides a categorical framework for describing and analyzing existing nonlinear magnification systems
- develops a transformation-based nonlinear magnification system that:

  - implements transformations with a RISC-type pipeline having simple modular components
  - is more expressive that existing systems
  - uses piecewise linear functions to efficiently approximate conventional continuous transformation routines
  - performs orders of magnitude more efficiently than published performance numbers for other systems with similar transformational functionality

- provides a mathematical foundation for the relation between magnification and transformation in more than one dimension
- introduces the implicit magnification field, which:
  - allows implementation-independent analysis and quantification of nonlinear magnification effects across widely different systems and transformations
  - provides consistent visual cues to enhance perception of nonlinear magnification effects
- develops the first magnification-based system for nonlinear magnification, that:
  - separates the magnification specification task from the mechanism used to produce the transformations, thus greatly simplifying the specification

task

  - allows fluid and shifting magnification specification without restrictions of discrete foci; transformational expressiveness goes far beyond what is possible with existing systems.
  - provides an iterative method for constructing a suitable transformation from a given magnification specification
  - serves as a basis on which to overlay customized nonlinear magnification interfaces, from low-level node-by-node specification to application-level constructions. two of these interfaces are particularly note-worthy:
    * *magnification brushing* to create subtle magnification effects in images
    * *data-driven magnification* allows properties of data to directly define the magnification field best suited for viewing that same data

- provides a framework for describing different levels on which nonlinear magnification techniques can be applied
- provides tools for synchronization of rendering detail functions with complex nonlinear magnification transformations, thus allowing tightly integrated detail-in-context visualizations
- introduces new applications of nonlinear magnification (in addition to the ones already mentioned)

- visualization of higher-dimensional clusters for data-mining
- multi-level image magnification allows seamless integration of different global "views" of the information, with different informational content at each level

1.3 Outline of the Dissertation

Chapter 2 will describe a pipeline-based approach to generating nonlinear magnification via simple transformation functions. Chapter 3 introduces theory and methods for nonlinear magnification fields, along with some applications. Chapter 4 will discuss issues involved with the application of nonlinear magnification, illustrated with specific example applications. Chapter 5 will provide pointers to related work, and Chapter 6 will summarize the conclusions.

2

Transformation-Based Magnification

The distinction between *transformation* functions and *magnification* functions (in the context of nonlinear magnification) was originally defined by Leung and Apperley in [LA94] and expanded upon in [KR96d]. The transformation function is the function that is used to perform the actual transformations, whereas the magnification function (which is the derivative of the transformation function) indicates the actual degree of magnification implicit in the transformation function. The relationship between transformation and magnification functions in more than one dimension is a non-trivial issue, which will be addressed in detail in Chapter 3. The focus of this chapter will be on *transformation-based* magnification – the manipulation of transformation functions to achieve the magnification effect.

The transformation-based approach to nonlinear magnification is the approach that is most commonly encountered in the literature, and includes a wide range of methods. Some of these will be mentioned here, and Chapter 5 will contain a fuller description of these and other systems.

In this chapter we will describe the construction of transformation-based magnification effects via a pipeline paradigm, similar to the rendering pipelines commonly encountered in the computer graphics literature [FvDFH90]. This pipeline provides a modular RISC-type approach to nonlinear magnification, reducing complex transformations into their constituent components, which can be easily manipulated and combined. By applying this *transformation pipeline* to a regular grid of points covering the source domain, we obtain a transformation grid T. Viewing T shows the effects of the transformation pipeline across the entire source domain. Much of the functionality described in this chapter was originally reported in [KR96d], although without describing the pipeline paradigm for integrating the constituent components.

2.1 Single Transformations

We begin our construction of the full transformation pipeline by first creating a pipeline for single transformations (i.e. only one center of magnification). A schematic diagram of the single transformation pipeline is shown in Figure 2.1. In later sections

we will use this single-transformation pipeline as a building block for more complex transformations. The remainder of this section will describe the construction of this single-transformation pipeline. We begin with a description of the *Warp Transform* process, which represents the core functionality for producing the nonlinear magnification transformation. Following this we will expand the scope to include the domain mapping processes for constructing constrained domains.

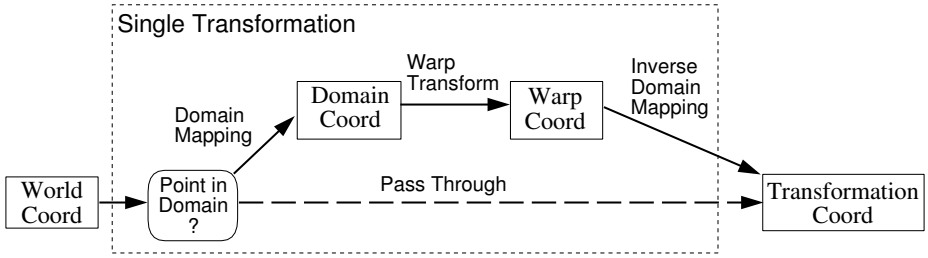


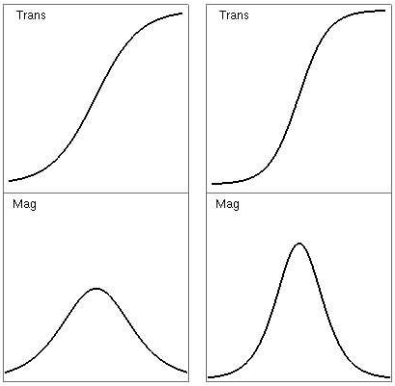
Figure 2.1: Pipeline for a Single Transformation. Point coordinates flow through this pipeline from World Coordinates to Transformation Coordinates. Each solid rectangular box represents a coordinate system, and the rounded rectangles represent an action to be taken. The transformation processes between coordinates systems are labeled edges.

2.1.1 1D Transformation Functions

The requirements for a 1D transformation function f for nonlinear magnification are straightforward: we require that f be C^0 continuous, monotone and that the range boundaries not exceed the domain boundaries in order to guarantee non-occlusion

(order preservation) over the entire domain. A desirable property for f is a smoothly varying slope over some domain such that some sections of the curve will have slope ratio > 1 (areas of magnification) and some other sections of the curve have slope ratio < 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 correspondingly decreasing the slope in other areas. These properties of 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 across the domain.

Several different choices exist for a 1D transformation function, and the hyperbolic tangent function $f(x) = tanh(x)$ provides a good example of a function which has the properties described above. $tanh(x)$ is well behaved across all domains ($\mathcal{R} \rightarrow [-1, 1]$), and there is no need to normalize the domain coordinates to some specific range before applying the magnification. To this transformation function we can add a parameter β which controls the degree of magnification, so that $f(x, \beta) = f(x\beta)$. A mechanism for moving the center of magnification across the domain is easily achieved: if we want x_0 to be the center of magnification, we simply replace $f(x)$ with $(f(x - x_0) + x_0)$ and the center of maximal magnification will translate to the desired location. Figure 2.2 shows the $tanh$ transformation function and its associated magnification function, alongside a version with modified β.

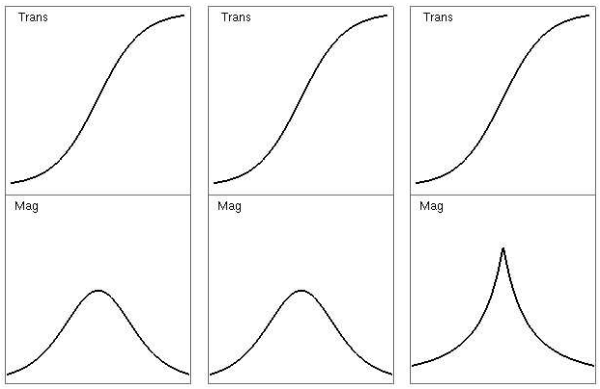


(a) $\beta = 2.0$ (b) $\beta = 4.0$

Figure 2.2: Tanh Transformation/Magnification

Other choices for a 1D transformation function are possible. Sarkar and Brown [SB94] use the *fisheye* function $G(x) = \frac{(d+1)x}{dx+1}$ to perform this transformation. $G(x)$ is well behaved and has desirable nonlinear 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. Another possibility is a modified logistic function ($f(x, \beta) = \frac{2.0}{(1.0+e^{-2\beta x})} - 1.0$) which is visually very similar to $tanh$, but is computationally less expensive on many machines [KR96d]. It will also be shown in section 2.5.1 that piecewise linear functions can be used to provide a reasonable approximation to these 1D transformation functions. Figure 2.3 shows a comparison of the fisheye, $tanh$ and logistic transformation functions. Throughout the rest of

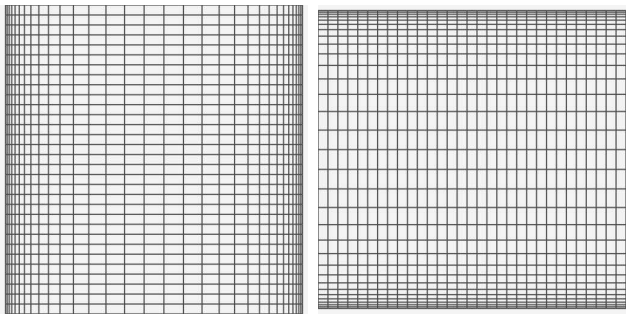
this chapter the *tanh* function will be used as the 1D transformation function, unless specified otherwise.



(a) *tanh* (b) logistic (c) fisheye

Figure 2.3: 1D Transformation Functions

As will be shown in the following subsection, 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. Figure 2.4 shows horizontal and vertical applications of 1D transformation functions. For another example, a study [KM96] details using using a 1D "magnification bar" for viewing highly structured text documents. This application is described in greater detail in Section 4.1.2.2.

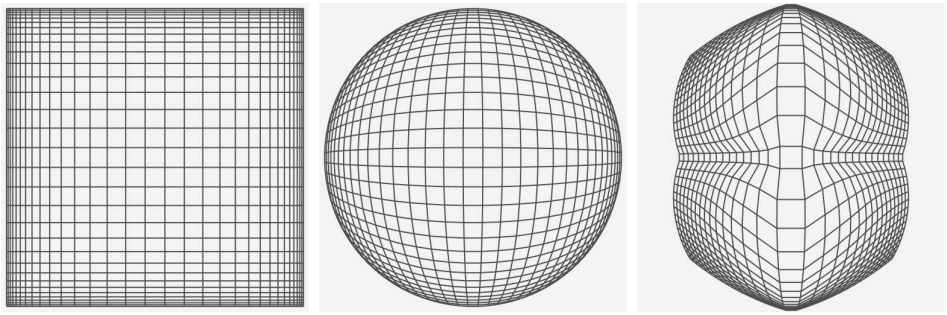


(a) horizontal (b) vertical

Figure 2.4: One Dimensional Transformations

2.1.2 2D Transformations

By applying these simple 1D transformation functions to 2D coordinate spaces in various ways, many different effects can be produced. In the discussion of the various methods below, H is the center point of the magnification and P is the point (x, y) to transform. Since all transformations are relative to H, it is also convenient to define $\hat{P} = P - H$ for use in some parametric equations. P' represents the transformed point. In our pipeline diagram for single transformations (Figure 2.1), these transformations are represented by the *Warp Transform* process. Figure 2.5 shows the effects of three such transformations, described below.



(a) orthogonal (b) radial (c) bi-radial

Figure 2.5: Two Dimensional Transformations

Orthogonal:

This 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. The transformed point is $P' = (x', y')$ where:

$$x' = H_x + f(\hat{P}_x, \beta_x) \tag{2.1}$$

$$y' = H_y + f(\hat{P}_y, \beta_y) \tag{2.2}$$

Here β is not a single parameter, but a vector representing degree of magnification in the x and y directions. This transformation is similar to the orthogonal stretching described in [SSTR93], preserving horizontal and vertical lines within

the domain, and allows for independent control of the magnification parameters for the x and y axes.

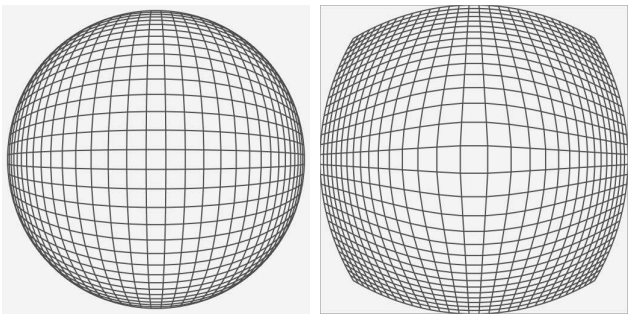
Radial:

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

$$r = \sqrt{\hat{P}_x^2 + \hat{P}_y^2} \tag{2.3}$$

$$P' = H + \frac{f(r, \beta)}{r}\hat{P} \tag{2.4}$$

Here r is the radius component of the polar coordinates of $\hat{P}$, and β is a scalar parameter which controls degree of magnification in all directions. This transformation preserves angles relative to the center of the magnification. Additionally, the fisheye effect is familiar to most users, and provides a ready analogy for the user to relate to. The result is the same as the radial and polar transformations described in [KS78, KRB94], except that the parametric formulation used here does not require computationally expensive trig functions for explicit polar-rectangular coordinate conversion. Figure 2.6 shows how using different 1D basis transformation functions affects the 2D radial transformation.



(a) *tanh* (b) fisheye

Figure 2.6: Different 1D Basis Functions for 2D Radial Transformation

Bi-radial:

Although the radial and orthogonal transformations are the most commonly used ones, there can be many other ways in which 1D transformation functions can be used to construct 2D transformations. One such way is the bi-radial transformation which 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* of the displacement is weighted by the x and y components of the distance from the center of magnification as in the orthogonal transformation.

$$r = \sqrt{\hat{P}_x^2 + \hat{P}_y^2} \tag{2.5}$$

$$r' = \frac{\hat{P}_x}{\hat{P}_x + \hat{P}_y} f(r, \beta_x) + \frac{\hat{P}_y}{\hat{P}_x + \hat{P}_y} f(r, \beta_y) \tag{2.6}$$

$$P' = H + \frac{r'}{r}\hat{P} \tag{2.7}$$

This transformation preserves angles relative to the center of magnification, and also provides some degree of independence for x and y magnification parameters. This can be used to advantage in some visualization tasks such as the World Wide Web navigation application described in Section 4.1.2.3. This application involves visualization of non-regularly spaced acyclic graph structures; the bi-radial transformation allows greater control of the view of this graph, and staggers the node labels so that they do not overlap as much as when the regular orthogonal transformation is used.

2.1.3 N-Dimensional Transformations

The extension of these techniques to three dimensions is straightforward; an extra z coordinate is added which is treated the same as the (x, y) coordinates. The simplicity of extension to higher dimensions distinguishes these techniques from many other nonlinear magnification systems which rely on perspective projections of 3D surfaces to achieve the magnification affect (such as Perspective Wall [MRC91], Document

Lens [RM93], 3D Pliable Surfaces [CCF95a] and Perspective Tunnel [MK97]). These perspective-based systems are *view dependent*; they rely on a physical viewing model in 3D space to create magnifications of 2D data. While perspective-based systems can take advantage of accelerated graphics hardware, a dimension is a terrible thing to waste. In contrast, the techniques presented here (among other systems) are *view independent*, and do not require a physical viewing model.

Although the math extends readily to higher dimensions, additional problems arise when we move to 3D information spaces. In particular, the problem of occlusion arises, and areas of magnification may be blocked from view. The occlusion problem is inherent to 3D visualizations in general, and there are many different techniques for dealing with it. Such techniques include transparency, clipping planes and recent work on distortion-oriented techniques [CCF96a]. There are some applications however where occlusion does not present as much of a problem.

One example of this involves visualization of high-dimensional clusters. Typically the data in these applications is composed of tight clusters of data in a sparse information space. Magnification of individual clusters might possibly cause occlusion of neighbouring data, but the sparsity of the data generally means that this will not be as much of a problem. An application involving such cluster visualizing for data-mining is described in Section 4.1.2.4.

2.1.4 Combined Linear/Nonlinear Transformations

An advantage of linear magnification is that it produces *distortion-free* zooming when the aspect ratio is maintained. For example, a user viewing textual data would prefer to see the letters at the area of maximal magnification without the distortions presented by nonlinear transformations. This is relevant when texture mapping [BN76] is used to fit arbitrary images (often containing text) to the transformed grid [KR96b]. As the grid points are transformed, the image is transformed along with them.

We can combine the advantages of linear and nonlinear magnification by linearly magnifying all points within a given region A, producing a magnified region A'. $\sim A$ is the region defined as the entire domain surrounding but not including A (i.e. the complement of A). We perform a nonlinear transformation on all points in $\sim A$, and then linearly interpolate the transformed points into the region surrounding A'. Figure 2.7 shows a schematic representation of this process.

Note that the maximum degree of magnification that can be used in the linearly magnified area is constrained by the size of the region A to be magnified (inversely proportional), and the size of the target region A' (directly proportional). In our pipeline schema (Figure 2.1), these combined linear/nonlinear transformations are

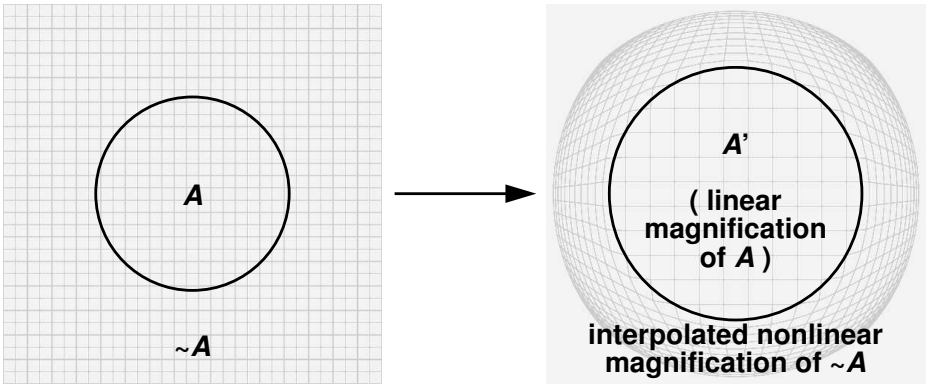


Figure 2.7: Schematic Diagram for Linear/Nonlinear Combinations

represented by the single *Warp Transform* process. The reason why this is not broken down into smaller stages in the pipeline will become evident in the upcoming discussion of 1D piecewise linear functions in Section 2.5.1, where we will see how this complex operation can be collapsed into a single step.

Figure 2.8 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 nonlinear magnification. In the right image, a rectangular area is used for the linear magnification and the surrounding points are transformed by an orthogonal nonlinear magnification.

Figure 2.9 shows how these linear/nonlinear combinations can be used to alleviate the distortion of magnified texture maps containing textual information. A fuller

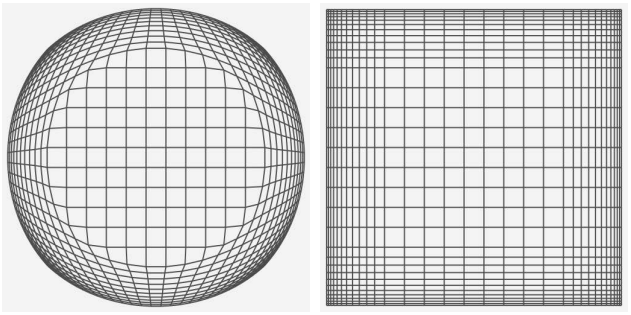


Figure 2.8: Combined Linear and Nonlinear Transformations

discussion of the use of texture mapping for image magnification will be provided in Chapter 4.

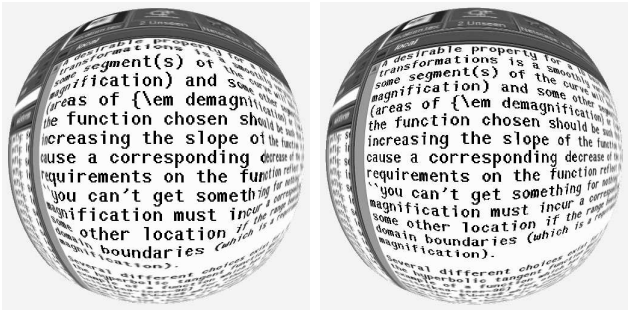


Figure 2.9: Linear/Nonlinear Image Magnifications Contrasted

2.1.5 Bounded Transformations

It is often 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-region of the domain, constraining the transformed points to the source sub-region.

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 2.10 shows examples of bounded domain transformations (note that the linear/nonlinear transformation is used *within* the constrained domain in the example on the right).

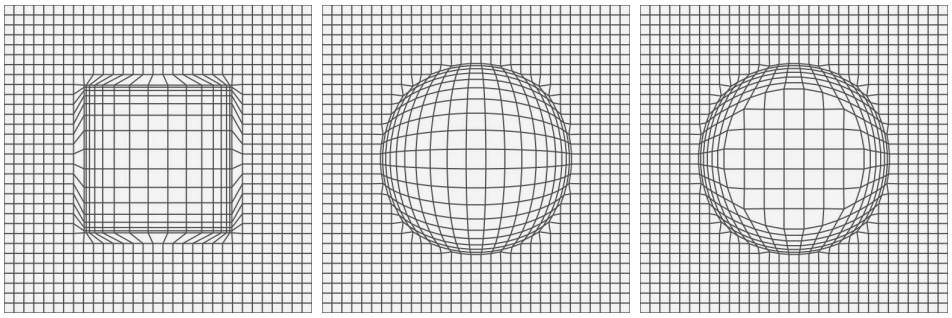


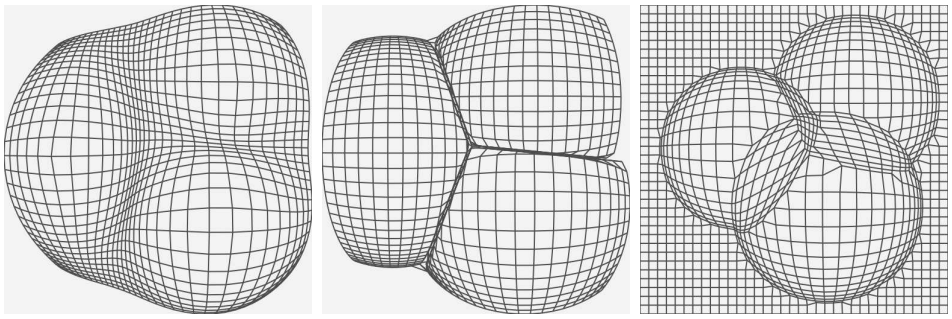
Figure 2.10: Bounded Transformations

The mechanism for performing these constrained domain transformations is similar to that used for the combined linear/nonlinear transformations described in Section 2.1.4, involving simple linear interpolations between regions. Here we perform the nonlinear transformation *inside* the region A, leaving all points outside of A untransformed. Note that it is often desirable to map the points inside A to some regular domain (e.g. $[-1,1] \times [-1,1]$) before performing the actual transformation on them and then re-interpolating the transformed points back into A. These linear mappings

are often easier to perform if we use a transformation function with a fixed range (such as $\mathcal{R} \rightarrow [-1, 1]$). In our pipeline diagram (Figure 2.1), the functionality for bounded transformations is indicated by the *Point in Domain?* test and the *Domain* and *Inverse Domain* mappings.

2.2 Compound Transformations

The earliest published method for nonlinear magnification (Polyfocal Projection [KS78]) included the ability to combine multiple centers of magnification simultaneously in a static environment. Many subsequent interactive systems did not have this capability, although more recently techniques have been developed to achieve such compound transformations in various ways. This section will consider several methods for combining multiple transformations. These techniques fall into three general categories, each with their own distinct questions and characteristics: 1) globally combining the transformations (averaging), 2) restricting the transformations so they do not overlap (partitioning), and 3) applying the transformations in succession (composition). The images in Figure 2.11 are the result of applying instances of these three approaches to a regular two dimensional grid, we describe each of these approaches below.



(a) average (b) partition (c) composition

Figure 2.11: Compound Transformations

2.2.1 Averaging

This is achieved by independently applying each transformation to a point in the original domain, and then making the final transformed point the average of the independent transformations. When the mean average is used, the results are similar to those from [KS78, MS91, KRB94]. If there are N transformations operating on a point P, and the effect of each transformation n is represented by f_n, then the average is straightforwardly given as:

$$P' = \frac{1}{N}\sum_{i=1}^{N} f_i(P) \tag{2.8}$$

Alternatively we can weight the transformed points inversely proportional to the distances between the centers of magnification (H_n) and the point in the original domain using the equation below (along with an additional test to ensure that $P \neq H_j$

for some $j\epsilon[1..N]$), so that the weights compensate for the reduction in magnification caused by the averaging. This enhances both degree and localization of the individual transformations. Figure 2.12 shows a comparison of mean and weighted averaging for compound transformations. The pipeline configuration for averaged transformations is shown in Figure 2.13.

$$P' = \frac{\sum_{i=1}^{N} \frac{f_i(P)}{||P - H_i||}}{\sum_{j=1}^{N} \frac{1}{||P - H_j||}} \tag{2.9}$$

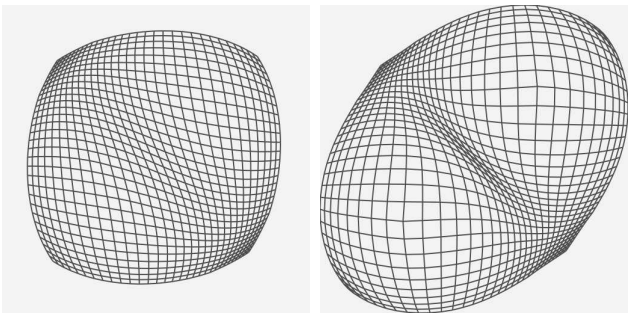


Figure 2.12: Mean and Weighted Average Compound Transformations

2.2.2 Partitioning

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

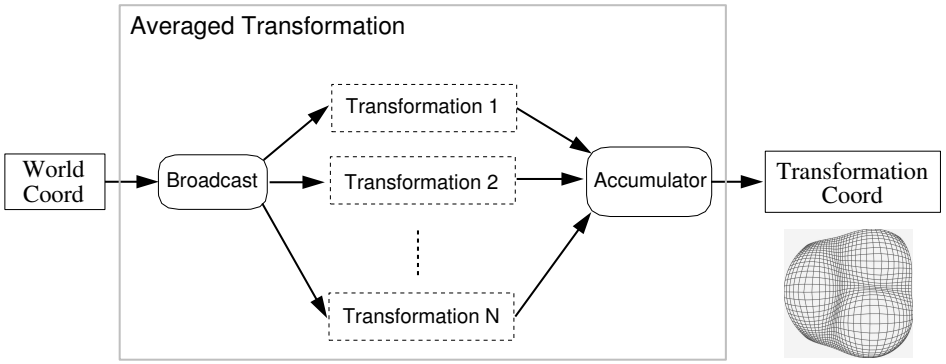


Figure 2.13: Pipeline for Averaged Transformations

has been performed, the point is clipped back along the ray between itself and the transformation center (if necessary) until the clipped point is again closer to the original transformation center than to any other transformation center. This produces a partitioning of the domain space similar to Voronoi diagrams from computational geometry, so that each transformation has its own "domain of influence". The visual result of this method is similar to that produced by the vector blending method in [CCF95a], except that with this system the boundaries between transformations are independent of the degree of magnification, and each region can be magnified or demagnified without affecting the partitioning of the space. The pipeline configuration for partitioned transformations is shown in Figure 2.14.

A method for achieving this effect works as follows: given N transformations, the effect of each transformation n with center of magnification H_n is represented by f_n.

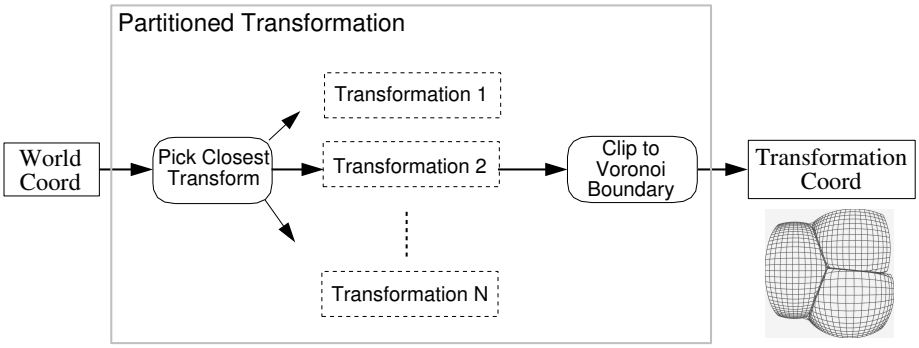


Figure 2.14: Pipeline for Partitioned Transformations

To transform point P, we first find the closest H to P and label it H_i (if more than one H is equidistant to P then choose the first one encountered). Then we perform the transformation of i on P so that $P' = f_i(P)$. Since P' might now be closer to some $H_j (j \neq i)$ than to H_i, we may need to pull P' back along the ray defined by $R = H_i + t(P' - Th_i)$. This is achieved by finding the distance between P' and each $H_j (j \neq i)$, if the distance is less than between P' and H_i, then the point is pulled back to the intersection point between R and the half-plane Voronoi boundary between H_i and H_j.

2.2.3 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 allows for this through an interface to a stack of transformations, and all transformations are applied to the domain in order from the bottom to the top of the stack. The effect of this (when magnifications with bounded domains are used) is that of a stack of lenses layered on top of the display, each of which can be controlled independently. The pipeline configuration for composition of transformations is shown in Figure 2.15.

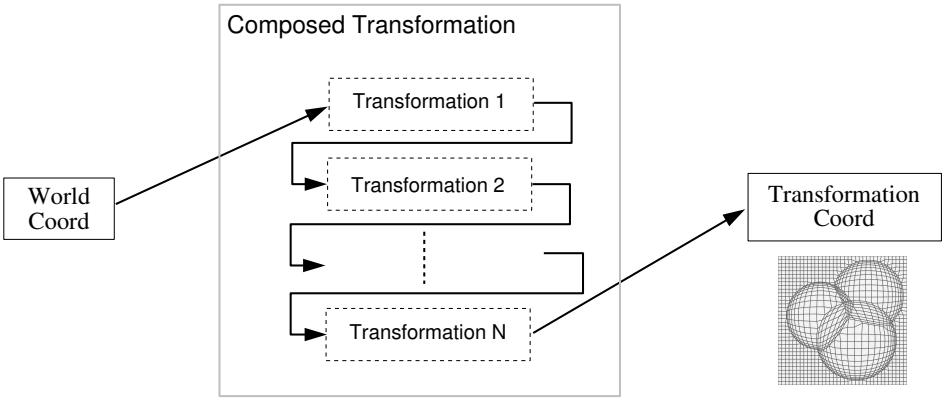


Figure 2.15: Pipeline for Composition of Transformations

2.3 Performance

Table 2.1 compares timing results for some of the typical transformations which are possible with the transformation pipeline that has been discussed to this point.

All timing results in this chapter were obtained on a MIPS 195 MHz R10000 processor. The peak transformation rate found here is 3.3 million transformations per second. This is fast enough to transform every pixel coordinate in a 640×480 display at an interactive 10.8 frames per second, although in normal use a much smaller grid of about 50×50 nodes is sufficient to generate typical transformations (which would be transformed at about 1300 frames per second). This represents a significant performance increase over published performance numbers for systems having similar transformational expressiveness (on the order of two orders of magnitude, see Chapter 5 for details). Later sections will discuss ways in which these performance results can be significantly improved still further.

Function	Time (10^{-6})	Normalized (10^{-6})	TPS(10^6)
Null	0.15	0.00	6.67
Orthogonal [2.5(a)]	0.81	0.66	1.23
Radial [2.5(b)]	0.80	0.65	1.25
Bi-Radial [2.5(c)]	1.34	1.19	0.75
Radial/Linear [2.8]	1.28	1.13	0.78
Bounded Radial [2.10]	0.30	0.15	3.33
Bounded Radial/Linear [2.10]	0.37	0.22	2.70
Averaged [2.11(a)]	4.69	4.54	0.21
Partitioned [2.11(b)]	2.91	2.76	0.34
Composition [2.11(c)]	1.79	1.64	0.56

Table 2.1: Transformation Pipeline Timings. Numbers in brackets are the corresponding figure numbers for the transformation. Times shown are average times (in seconds) for computing a transformation on a single point. Normalized times reflect the difference between the time for the transformation function and a "Null" transformation which returns the original point, thus eliminating overhead common to all transformations. TPS is the number of points transformed per second (Transformations Per Second).

2.4 Global Operations

The final stages of the transformation pipeline affect global properties of the transformation grid T. These finishing stages ensure that the transformed data points still fit within the viewing area, and also help to preserve the spatial ordering of the transformation grid. These final operations are shown within the transformation pipeline framework in Figure 2.16.

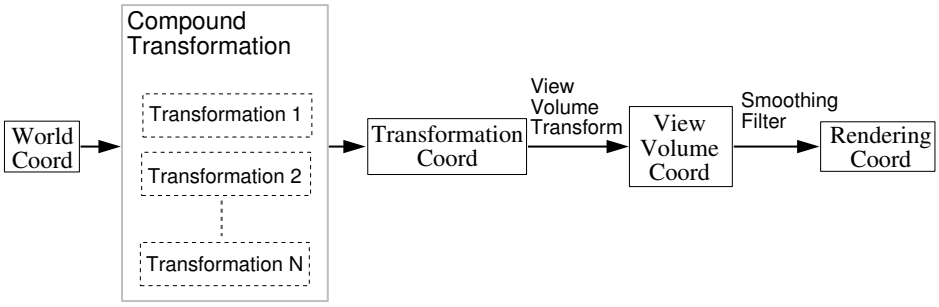


Figure 2.16: Global Pipeline

2.4.1 View Volume Operations

A problem with these magnification techniques can arise when the magnification transformation "pushes" magnified data outside of the normal viewing volume (in 2D this means pushing beyond the window boundaries). There are several methods for dealing with this problem which we discuss here for the 2D case, and illustrate in

Figure 2.17.

Crop:

This is the simplest method where every point that falls beyond the window boundaries is simply cropped and not rendered it at all.

Constrain:

Here the transformed data are checked to see if they fall beyond the window boundaries. If some of the data fall outside the boundary, then the window domain is extended to incorporate all of the data. Alternatively, the data points can be rescaled to fit within the window.

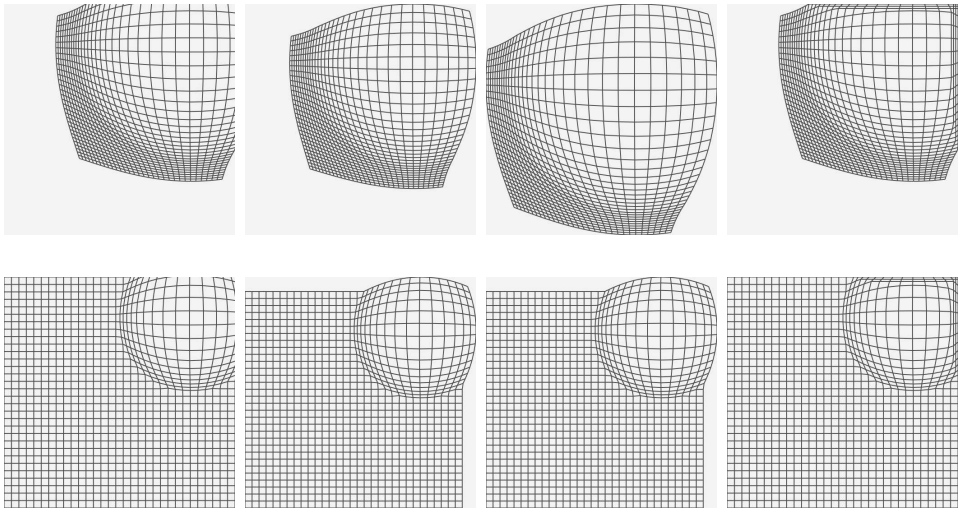
Maximal:

This uses techniques similar to the previous method to ensure that the boundaries of the transformed data points extend to exactly the window boundaries.

Clip:

The major disadvantage of the constrained and maximal window boundary methods is that for bounded transformations the global context no longer remains static, but instead is compressed to remain within the window boundary when the magnification moves near the edge of the window. An alternative is to clip points beyond the window boundary to the window boundary itself. This

will result in a line of overlapping points, which can be smoothed out using the filtering techniques described in the next subsection.



(a) crop (b) constrain (c) maximal (d) clip

Figure 2.17: View Volume Operations for Infinite and Bounded Domains

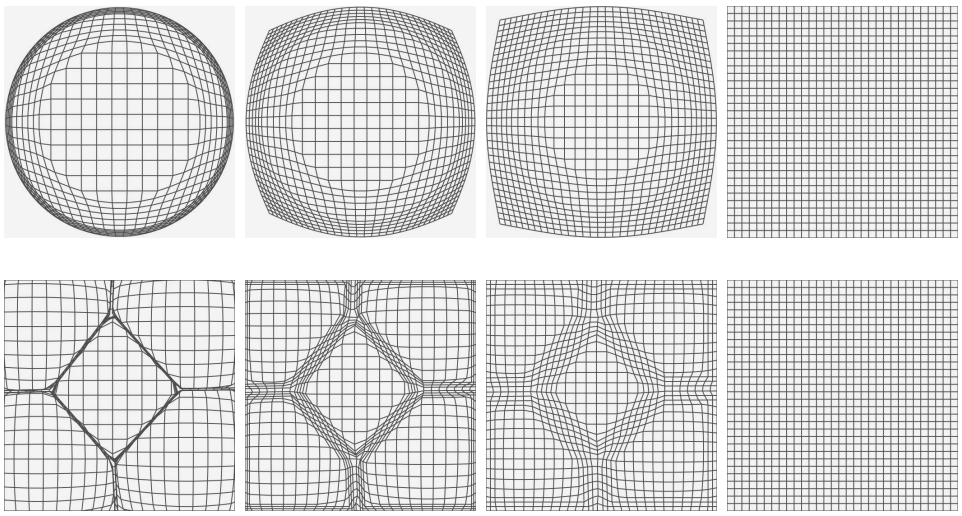
2.4.2 Smoothing Filter

Independent of the complexity of the transformations employed, it is often useful to have a *single* parameter for controlling the degree to which *all* transformations take effect. This can allow the user to smoothly shift between the warped and unwarped views of the space. Such functionality can be provided through a simple filter applied to the transformed and untransformed points, which provides a variable weighting of

the points. Let s be the weight attached to the source point P, and d be the weight attached to the transformed point Q. We make the conditions that $0 \leq s \leq 1$ and $d = 1 - s$. The filtered point is then $sP + dQ$. Figure 2.18 shows examples of changing the single filter parameter to alter the overall effect of 2 different complex transformations. This simple method provides the user with an effective means of realizing the relationship between normal and distorted views. In addition, this technique can greatly facilitate construction of mappings such as the linear/nonlinear combinations described in Section 2.1.4, the bounded transformations of Section 2.1.5, the partitioning method for compound transformations in Section 2.2.2 and the view volume operations of Section 2.4.1; points can be mapped to the extreme boundaries of a region, and the filter will enforce the original ordering of the points, thus smoothing out the spacing between them.

2.5 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 significant performance gains and also allow for a more expressive class of transformations (because of the ease of manipulation).



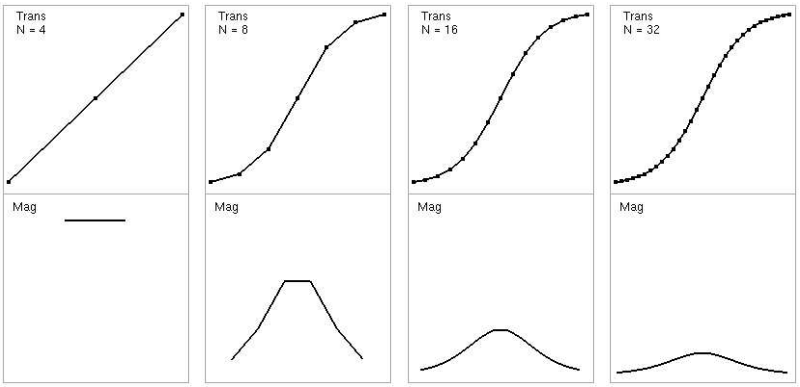
(a) $s = 0.0$ (b) $s = 0.3$ (c) $s = 0.6$ (d) $s = 1.0$

Figure 2.18: Smoothing Filter for Radial and Compound Transformations

2.5.1 1D Piecewise Transformations

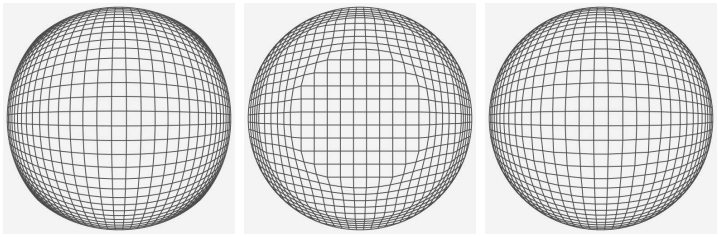
These transformations involve piecewise linear approximations of the single-variable transformation functions which are used to drive the higher-order (2D and 3D) transformations (as described in Sections 2.1.2 and 2.1.3). As an example of this, a piecewise approximation of $tanh(x)$ can be used to replace that computationally expensive 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. Figure 2.19 shows piecewise linear approximations to the $tanh$ transformation function, along with the associated magnification functions. Table 2.2 shows some timing results for the simple radial transformations shown in Figure 2.20,

comparing the *tanh* and piecewise linear transformation functions.



(a) $n = 4$ (b) $n = 8$ (c) $n = 16$ (d) $n = 32$

Figure 2.19: Piecewise Linear Approximations to *tanh*. (n refers to the number of magnification values, by convention there are 2 magnification values of zero – one on each end of the transformation – that are not shown in these examples.)



(a) *tanh* (b) $n = 8$ (c) $n = 16$

Figure 2.20: Radial Transformation with *tanh* and Piecewise Linear Approximations

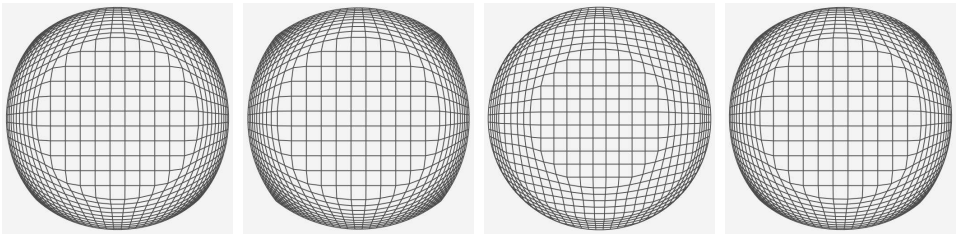
Function	Time (10^{-6})	Normalized (10^{-6})	TPS(10^6)
Null	0.15	0.00	6.80
tanh	0.80	0.65	1.25
logistic	0.79	0.64	1.26
fisheye	0.59	0.44	1.71
1D Piecewise $n = 8$	0.56	0.41	1.79
1D Piecewise $n = 16$	0.56	0.41	1.79
1D Piecewise $n = 32$	0.56	0.41	1.79

Table 2.2: Timings for Radial Transformation. (TPS = Transformations Per Second).

Note that Table 2.2 shows the time for the piecewise approximations to be just slightly faster than the fisheye transformation time. While the expressive power of the fisheye basis transformation is limited to altering the degree of curvature of the function based on a single parameter, piecewise functions can transform points with 1D transformations of arbitrary complexity at no extra computational cost, and there is no time increase for the piecewise function as the number of components in the approximation is increased.

We can see a simple example of the additional expressiveness of the piecewise transformation by considering the methods used for producing the combined linear/nonlinear magnification discussed in Section 2.1.4. 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 2.21 shows a comparison of the results of these transformations (the piecewise version was obtained by using the piecewise radial transformation with 6 segments). Table 2.3

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 2.2. These data clearly show a performance potential for the piecewise transformation as transformation complexity increases.



(a) *tanh* (b) fisheye (c) 1D (d) 2D

Figure 2.21: Comparison of Combined Linear/Radial Transformations

Function	Time (10^{-6})	Normalized (10^{-6})	TPS(10^6)
tanh	1.28	1.13	0.78
fisheye	1.06	0.91	0.94
1D Piecewise	0.57	0.42	1.77
2D Piecewise	0.62	0.47	1.63

Table 2.3: Timings for Combined Flat/Radial Transformation

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). In order to enforce non-occlusion, we also require the function to be monotonically increasing. The degree of magnification provided by the function is directly proportional to the slope of the segment at that point. There are several ways in which such functions can be constructed. The simple piecewise functions used in this paper were all generated through 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 a 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; as 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 enforcing monotonic ordering of the segment end points, or by using spline functions with higher degrees of continuity 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 assigning 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 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. If only positive magnification values can be specified, the integration will also produce a monotonically increasing function. As with the transformation functions, this process is also amenable to the use of spline functions for direct construction.

2.5.2 2D Piecewise Transformations

In the previous subsection, we explored methods for using piecewise linear functions to approximate the 1D transformation functions described in Section 2.1.1. In this subsection, we will examine a more powerful set of piecewise transformations, where an approximation is made of a complete two dimensional transformation.

The most straightforward method to construct a 2D 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 controlled by parameterizing the resolution of the "sampling grid" that is used.

Table 2.3 shows that even for *slightly* complicated transformations such as the Flat/Radial combination (Figure 2.21), construction of a 2D piecewise transformation can result in significantly faster transformation times than for the procedural fisheye technique. As with the 1D 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 1D piecewise approach is the fastest of the three, requires much less memory than the 2D, and would be a better choice for this particular transformation.

When we moved from continuous basis transformation functions to their 1D 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 2D 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 2.2) and express them in terms of a single 2D 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 2D piecewise transformation grid. This effectively allows us to collapse the entire transformation pipeline into a simple table lookup operation. The timing results shown in Table 2.4 indicate a magnitude of order increase in performance when comparing the procedural (fisheye) and 2D piecewise approaches to producing the transformation shown in Figure 2.22.

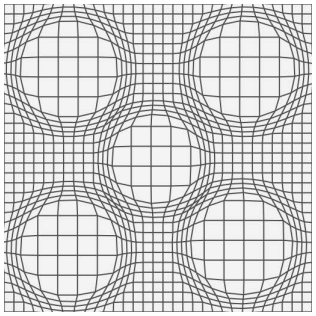


Figure 2.22: Compound Transformation for Table 2.4

Function	Time (10^{-6})	Normalized (10^{-6})	TPS (10^{6})
Fisheye	4.97	4.82	0.20
1D Piecewise	4.65	4.50	0.22
2D Piecewise	0.61	0.46	1.65

Table 2.4: Times for Compound Transformation in Figure 2.22

Direct user manipulation of 2D piecewise continuous functions poses more of a problem than is the case with 1D piecewise functions. The transformation grid can be manipulated directly by the user, although additional tools would likely be required to make the task less complex than individually setting each node on the grid. Direct

manipulation of the magnification values raises a whole set of new problems for dealing effectively with the 2D integration to find the associated transformation grid. The following chapter will address these issues in detail.

Because of the overhead involved in construction of 2D piecewise transformations, they are best suited to tasks which are dynamic in time and placement rather than in shape. Once computed, these 2D 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, and therefore 1D piecewise or continuous functions might be better suited to those applications where the shape of the transformation changes frequently.

2.6 Coda

This chapter has described a transformation pipeline for nonlinear magnification, and shown how simple modular components can be combined to produce complex magnification effects. Piecewise linear approximations can be used to reduce individual or multiple operational components into simple table lookup operations. As will be discussed in Chapter 5, this transformation pipeline provides much faster performance than is described in any other similar system for nonlinear magnification. The

core concepts of this system have been adopted by other researchers as a basis for the design of similar systems for nonlinear magnification, particularly in the application of collaborative geographic information system browsers [CPC97].

The transformation-based approach to nonlinear magnification is by far the most prevalent among the systems described in the literature. Although these systems encompass a high degree of variation in the methods used, they all have two characteristics in common. The first characteristic is the use of 2D transformations to stretch and compress regions of the domain, producing implicit magnification of the domain as a side effect of the transformations. The second characteristic is the use of explicit centers of magnification to define regions of magnification, thus constraining the interface to something which could perhaps be metaphorically described as a set of "nonlinear magnifying lenses".

In the next chapter, we will examine new methods for *magnification-based* nonlinear magnification, in which magnification can be expressed more directly, and which removes the requirement of explicit centers of magnification, instead allowing fluidly-shifting fields of magnification values.

3

Magnification-Based Magnification

The focus of the previous chapter was to define a nonlinear magnification system using simple modular components joined together to create complex magnification effects. Although this system for nonlinear magnification transformations was successful in reducing complex magnification effects to a set of easy to understand and implement operations, there are some general limitations which the system still has in common with existing nonlinear magnification systems.

The first major limitation of the existing nonlinear magnification systems could be referred to as the "tyranny of the foci". Although explicit centers of magnification are clearly desirable in many cases, this also puts severe limitations on the types of magnification and interaction which can be produced. Interaction becomes an exercise in manipulation of discrete magnifying "lenses", and additional expressiveness of magnification comes primarily through additional and/or complex lenses, thus increasing

the computational cost of computing the overall transformation.

A second difficulty encountered with existing systems involves determining the overall effect of complex transformations. A general purpose mechanism would be useful to determine the effects of complex transformations with multiple foci, so that the global effect of the transformations can be determined across the entire space of a domain, rather than just at the centers of magnification or other discrete points.

Thirdly, all of the existing nonlinear magnification systems closely tie the method for specifying the magnification to the method used to produce the transformation. In most cases the magnification is only a side-effect of specifying a transformation. These interdependencies create complications for each method. A system in which the magnification values can be specified *independently* of the transformation mechanism would provide more direct control of the specification, as well as allowing the magnification values to be analyzed, combined and manipulated in a more systematic fashion.

In this chapter we will develop further a theory of nonlinear magnification that addresses these issues. In particular, we reduce the concept of nonlinear magnification to a field of scalar magnification values. Broadly speaking, these nonlinear magnification fields provide benefits at two levels. First, they serve as a basis for analyzing the effects of existing techniques, even though their underlying mechanisms may be very different. Second, they directly define space transforming visualizations

and can be manipulated in computationally efficient and conceptually effective ways, thus yielding powerful visualization tools.

Defining magnification as a field of arbitrary scalar values provides a much greater expressiveness of magnification and ease of manipulation than is possible using other techniques. By removing the restrictions of discrete foci we allow fluidly shifting magnifications of arbitrary complexity, and can factor out magnification complexity from the time required to compute suitable transformation functions so that computation is *independent* of the complexity of the magnification function. We will look at a number of novel ways in which the flexibility of these nonlinear magnification fields can be used to create effective visualizations. The techniques presented range from low-level precise specification of magnification, through the creation of expressive user-interface techniques, to sophisticated magnification fields constructed by application programs. Many of the results described in this chapter are based on work that was initially reported in [KR97a, KR97b].

This chapter has several major sections. First we will develop a formal specification for the relationship between transformation and magnification. Then we will see how the implicit magnification field for a given transformation can be computed. Following this we will investigate an iterative method for computing a transformation from a given magnification field specification, and also examine some performance aspects of that method. Finally we will look at some of the ways in which interfaces

can be layered on top of these low level magnification fields.

3.1 Magnification Fields

When describing nonlinear magnification systems, it is useful to distinguish between *magnification* and *transformation* functions, as first described for the one-dimensional case in [LA94]. The transformation function directly stretches and compresses the space, while the magnification function (which is the derivative of the transformation function) represents the magnification values which are implicit in the transformation function. This distinction is central to the work described in this chapter. Converting between magnification and transformation functions in one dimension is a relatively straightforward task, however the situation is much more complicated in two or more dimensions. In the next two sections we will examine techniques for accomplishing these conversions, after introducing some basic definitions and notation. Although this chapter presents results in 2 dimensions, the view-independent nature of the techniques presented here allows for trivial extension to 3 or more dimensions.

Before exploring the mathematics for determining the local 2D magnification, it is illustrative to look at the simple 1D case, where the the relationship between transformation and magnification functions was first established by Leung and Apperley

in [LA94]. In particular, they identify magnification as the derivative of the transformation function. For a transformation function $y = f'(x)$, the derivative is given as:

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{3.1}$$

$$= \frac{\text{range length}}{\text{domain length}}$$

Thus for the 1D case, local magnification is computed as the limit of the *ratio of lengths*. A diagram of this relationship is shown in Figure 3.1. We will extend this result to two dimensions, where instead of ratio of lengths, we will use the *ratio of areas* to define magnification.

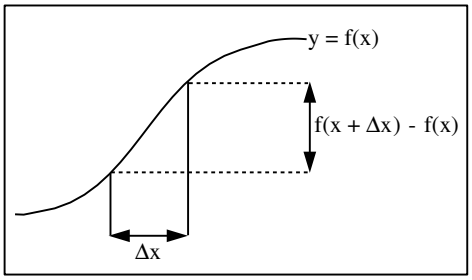


Figure 3.1: Transformation and Magnification Relation in 1D

Any magnification employs a transformation function t which moves points of a rectangular domain $\mathcal{D}$ within a frame ($(u, v) = t(x, y)$). t is a vector function which

can be also be expressed as:

$$t(x,y) = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} t_u(x,y) \\ t_v(x,y) \end{pmatrix} \tag{3.2}$$

Since we want the magnification to be non-occluding, we require that t is at least C^0 continuous, one-to-one and order-preserving (given $(u_1, v_1) = t(x_1, y_1)$ and $(u_2, v_2) = t(x_2, y_2)$, $x_1 < x_2$ implies $u_1 < u_2$, and similarly for y). For computational purposes, we deal with t only on a $p \times q$ integer grid $\mathcal{G}$, with $g : \mathcal{G} \to \mathcal{D}$ (g maps the regular grid $\mathcal{G}$ into the domain $\mathcal{D}$), and represent a discrete approximation of t with a quadrilateral grid T ($T = t \circ g$, where '$\circ$' represents composition). Given a region $\mathcal{R}$ in $\mathcal{D}$, we represent the transformation of $\mathcal{R}$ via t as $t(\mathcal{R})$ (also called the *image* of $\mathcal{R}$). This transformation is illustrated in Figure 3.2.

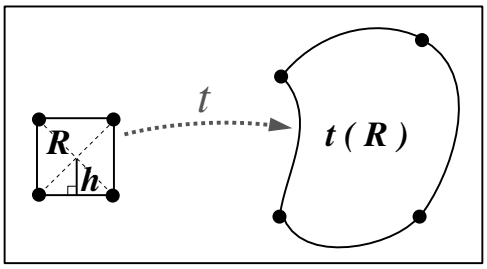


Figure 3.2: Nonlinear Transformation of a Region

A magnification field m is a 2D scalar field of the form $z = m(x, y)$ which gives the regional expansion caused by t around a point. As with t, m is normally represented

on $\mathcal{G}$ by a quadrilateral mesh M ($M = m \circ g$).

Informally, we would like our measure for magnification to represent the intuition that magnification in the real world is "how much bigger things get". A single scalar value indicating this would be readily analogous to the "power" commonly used for describing lenses. The mathematics that we use to define this magnification should serve to reflect this intuitive understanding.

The most obvious approach to measuring "how much bigger things get" is to compare the area of a region before and after the transformation has been applied and to make magnification the ratio of those areas:

$$\text{magnification} = \frac{\text{Area}(t(\mathcal{R}))}{\text{Area}(\mathcal{R})} \tag{3.3}$$

This is a direct extension of the definition of magnification in one dimension that provides a single scalar value reflecting the intuitive notion of magnification. We can formulate this area-based approach more precisely in terms of limits by creating an *area-based derivative* as defined below.

$$\text{magnification} = \lim_{h \to 0} \frac{\text{Area}(t(\mathcal{R}))}{4h^2} \tag{3.4}$$

By defining magnification in this way however, we can no longer assert that magnification is the true derivative of transformation in 2D. In fact, a true derivative of a 2D transformation function would yield multiple scalar values rather than a single magnification value. Although it would nice if the derivative relationship carried over directly from the 1D case, there are two major reasons why doing so would not provide an appealing definition for magnification:

- It does not match "real world" perception of magnification as a single scalar value.
- It does not reduce complexity of magnification specification in comparison to transformation-based approaches. It will be shown throughout this chapter that having a single scalar value for magnification greatly facilitates manipulation.

It should also be noted that by defining magnification as a single scalar value in this way, we lose some information about the original transformation. This means that given a transformation grid T, if we compute the magnification M and then construct a transformation grid T' from M, we may not end up with identical T and T'. In general, there may be many possible transformations for a given magnification specification. We will see however, that despite this loss of information about the original grid, good results can be obtained by constructing a transformation having the same intrinsic magnification properties.

Related to this issue, the area-based measure for magnification that we are using assumes that magnification is uniform locally and does not take into account scaling differences in the x and y directions, so that when scaling a region by d in the x direction and by $\frac{1}{d}$ in the y direction we will sometimes obtain a magnification value of 1 (depending on the specific area function being used). Although this may present some problems with the local definition of magnification, we will see in Section 3.3 that the implicit gradient of magnification values within the magnification field can provide additional information about such cases when they are part of a larger region that is non-uniformly scaled.

Now that we have defined magnification as a ratio of areas, we must also define how to compute the area of a nonlinearly transformed region $\mathcal{R}$ in order to find that ratio. In the following section we will calculate an approximation of these areas numerically, here we present the mathematical formulation for the continuous case. We can compute the area of $t(\mathcal{R})$ with the double (area) integral:

$$Area[t(\mathcal{R})] = \iint_{\mathcal{R}} dA_{xy} \tag{3.5}$$

A theorem showing how to find this integral for a nonlinearly transformed region can be found in Chapter 11 of [PM64], where it is used to develop the Change of Variables

Theorem. The formulation for the solution of this integral is:

$$Area[t(\mathcal{R})] = \iint_{\mathcal{R}} \left| \mathcal{J}\left(\tfrac{u,v}{x,y}\right) \right| dA_{xy} \tag{3.6}$$

There are a number of conditions which must be met for this integral to exist: t must be one-to-one and continuously differentiable over the domain $\mathcal{D}$, $\mathcal{R}$ must be a closed bounded region in $\mathcal{D}$ having some area, and the Jacobian $\mathcal{J}$ of t must never be zero (i.e. the transformation must be invertible).

Given the integral in Equation 3.6 for finding the area of a nonlinearly transformed region, we can now compute the local magnification as the ratio of the areas of the transformed and original regions:

$$\begin{aligned} m(x,y) &= \lim_{A_{xy}\to 0} \frac{\iint_{\mathcal{R}} \left| \mathcal{J}\left(\tfrac{u,v}{x,y}\right) \right| dA_{xy}}{A_{xy}} \\ &= \frac{\text{range area}}{\text{domain area}} \end{aligned} \tag{3.7}$$

In the following section we will approximate this limit using an area-based function m_c which computes the local magnification for each node in T. For computational efficiency, the area-based magnification we use here actually corresponds to the square of the linear "power" sometimes used in describing lenses.

3.2 Transformation Grid to Magnification Field Conversion

Conversion from a given transformation grid T to a magnification mesh M involves numerically approximating an area-based derivative of T. The computation begins with an area function a_t which, for each node in T, returns an approximation of the area defined by the neighbouring nodes. One possibility for this function simply uses the bounding box of the 4-connected neighbours $\{T(i+1,j), T(i-1,j), T(i,j+1), T(i,j-1)\}$. If the distance between node $T(i,j)$ at coordinates (x,y) and one of its 4-connected neighbours in an untransformed grid is h, then this area can be defined as (where "$\cdot$" represents the dot product):

$$a_t(x,y) = |(t(x-h,y) - t(x+h,y)) \cdot (1,0)| \times |(t(x,y-h) - t(x,y+h)) \cdot (0,1)| \tag{3.8}$$

We define C_a to be the constant area associated with any $T(i,j)$ in the untransformed uniform sampling grid:

$$\begin{aligned} Ca &= |((x-h,y) - (x+h,y)) \cdot (1,0)| \times |((x,y-h) - (x,y+h)) \cdot (0,1)| \\ &= 4h^2 \end{aligned} \tag{3.9}$$

The approximate magnification value for a point $T(i,j)$ is then given by $M(i,j) = m_c(g(i,j)) = m_c(x,y) = a_t(x,y)/C_a$. As the mesh resolution approaches infinity, this becomes:

$$m_c(x,y) = \lim_{h \to 0} \frac{a_t(x,y)}{4h^2} \tag{3.10}$$

More accurate area calculations are possible, such as explicitly finding the area of the four surrounding cells. In practice however, this increase in accuracy does not normally change the results significantly. Coarser approximations are adequate, as long as the area function is used consistently throughout the system.

Figure 3.3 shows an example of a transformation grid and its associated magnification mesh calculated with this method. This technique allows any nonlinear magnification system to create a landscape representation of its implicit magnification values. Elevation-based shading can be used to provide visual cues to the magnification level simply by mapping elevation values into a color/intensity ramp. This is in contrast to perspective based systems such as perspective wall [MRC91], where elevation and magnification do *not* correspond directly, but are also dependent on viewing parameters.

Figure 3.4 shows another example, this time illustrating multiple bounded regions and linear magnification; the transformation grid was generated using the techniques

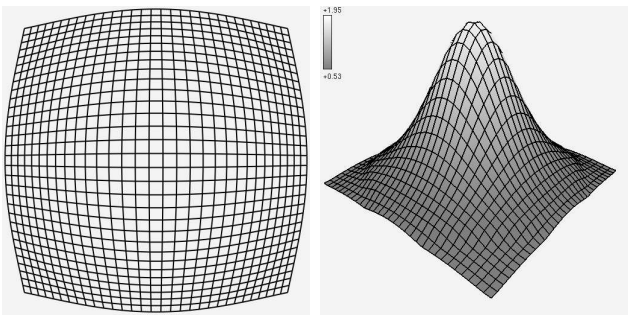


Figure 3.3: Transformation Grid and Implicit Magnification Field

described in [KR96d]. As a further example of how these techniques can be used to determine the implicit magnification generated by existing systems, we use the example of Perspective Wall [MRC91], which is representative of the class of nonlinear magnification systems that are based on a perspective projection of 3D surfaces (other examples include [RM93, CCF95a, MK97]). By sampling a perspective wall transformation function with a regular grid, we obtain a transformation grid which is used to generate the associated magnification mesh (see Figure 3.5). These examples show how transformation and magnification functions can now be tightly coupled across entire domains even for complex transformations. This method is very general-purpose in nature, and can be readily applied to any spatial nonlinear magnification system.

As an example of how implicit magnification fields can be used, consider the task of rendering node sizes proportional to the magnification produced by a complex transformation. Proximity to the centers of magnification is not an effective measure on

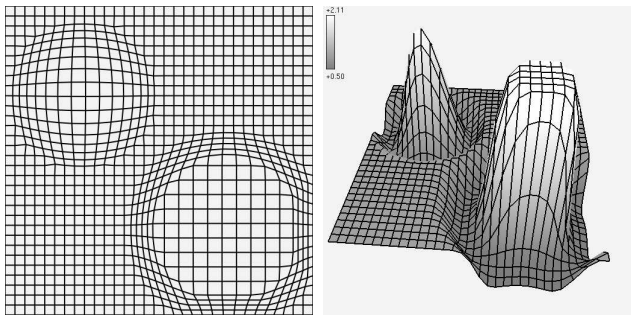


Figure 3.4: Complex Magnification Field

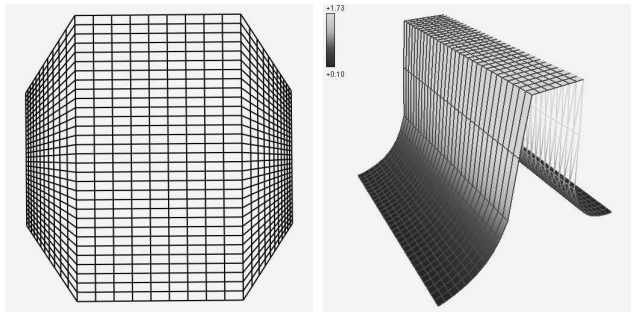


Figure 3.5: Perspective Wall Magnification

which to base node size, as it does not take into account the effects of multiple transformations and domain or smoothing parameters. By sampling the region around a node however, we can determine the local implicit magnification for that node by finding out whether that region expands or contracts. By sampling uniformly over a space, we create a magnification mesh and can then interpolate the magnification values between nodes to create a continuous magnification field. Any object can be rendered at an appropriate size or level of detail by finding its magnification value in the field. Figure 3.6 shows a simple example where the implicit magnification field is used to determine the size at which to render objects located within a complexly

transformed space. Further examples of similar ways in which the implicit magnification field can be used to determine the "detail" with which to render data objects are described in Chapter 4.

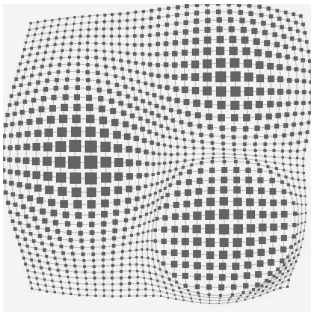


Figure 3.6: Node Size Proportional to Implicit Magnification

3.3 Magnification Field to Transformation Grid Conversion

While it is a relatively straightforward task to find the implicit magnification mesh M associated with a transformation grid T by computing the area-based derivative, it is a much more complex task to construct a suitable transformation grid given a magnification mesh. Simple perspective projections of these meshes from an aerial viewpoint are not effective because they may introduce problems of occlusion, as shown in Figure 3.7 (which shows a perspective projection of the magnification field in Figure 3.4).

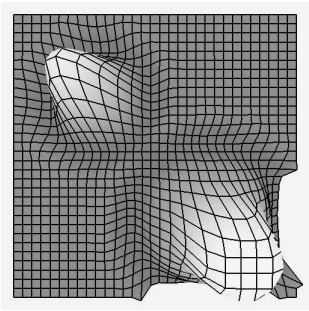


Figure 3.7: Occlusion from Perspective Projection

In general terms, we want to use the magnification mesh values to construct an order-preserving transformation grid having the same implicit magnification values. There are a number of issues which make this a difficult task. The most fundamental problem involves finding a meaningful way to convert a *single* magnification value into a *two* coordinate (x, y) transformation; there are usually many transformations possible for a given magnification. In general, the transformation to magnification conversion is inherently lossy, and hence the result of the magnification to transformation conversion is not uniquely defined. There exist direct methods for solving this problem, but these methods are unsuited to the specific task of generating nonlinear magnification transformations for visualization. Some of the major shortcomings observed with the direct approaches are:

- bounded regions of magnification in M should produce bounded regions of transformation in T to preserve a static context
- the transformation should be symmetric and centered around magnification

maxima, and not constructed relative to some arbitrary boundary of the domain

- solutions providing only correct area in T do not preserve desired visual properties of the magnification, such as scale and aspect ratio within regions of linear magnification

We can illustrate the shortcomings of these direct methods with a specific example. Figure 3.8 shows the effect of a bounded linear/radial transformation (produced using the methods from Chapter 2) on a checkerboard pattern, along with the associated implicit magnification field (both with elevation shading, and with the checkerboard overlayed). This will be our input specification for demonstrating the shortcomings of the direct methods. Note that the actual resolution of the underlying grid is much finer that the resolution of the checkerboard pattern in these examples.

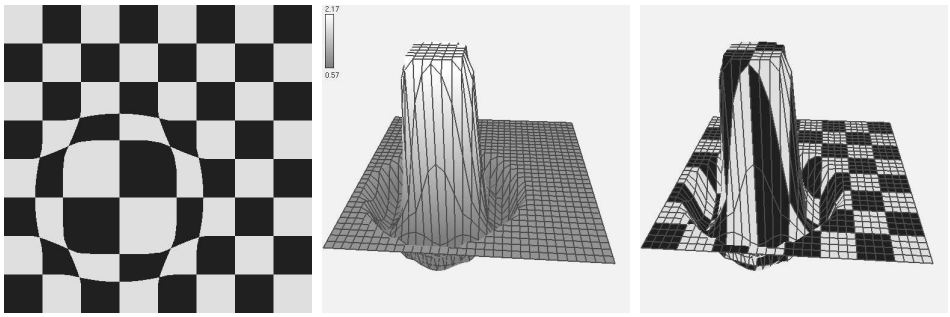


Figure 3.8: Specified Transformation with Bounded and Linear Magnification

The first direct method involves building up a transformation grid cumulatively by starting at one corner of the grid, and adding triangles to the grid on a row by

row basis, one pair of triangles at a time. The location and size of each triangle can be determined using simple trigonometry, although care must be taken to correctly deal with the acute angles and cumulative floating point errors which quickly develop as the algorithm progresses. Figure 3.9 shows the result of running this algorithm on the specified magnification from Figure 3.8. Although the solution is numerically correct (the window size has been enlarged to encompass the new boundaries), the result suffers from all of the previously mentioned shortcomings and does not present an effective visualization.

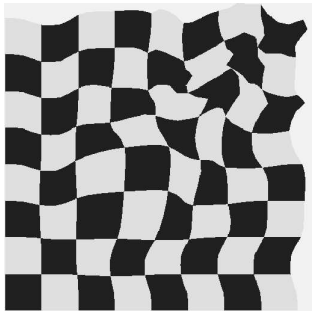


Figure 3.9: Cumulative Solution to Specified Magnification in Figure 3.8

The second method is based on histogram equalization techniques [GW93]. For the case of 1D magnification to transformation conversion, it is a simple matter to use histogram equalization to translate the magnification values into suitable transformations. For a specification having n magnification values $\{z_1, z_2, \ldots, z_n\}$, we can compute a 1D transformation having the desired implicit magnification over grid

points $\{x_0, x_1, \ldots, x_n\}$ with boundaries $(x_{\min}, x_{\max})$ using the equations:

$$\begin{aligned} x_0 &= x_{\min} \\ x_i &= x_{i-1} + \frac{z_i}{\sum_{j=1}^{n} z_j}(x_{\max} - x_{\min}) \quad (\text{for } i = \{1 \ldots n\}) \end{aligned} \tag{3.11}$$

We would like to be able to extend this to 2D by first applying this technique to the rows of the mesh to compute the transformed x values, then resampling the magnification values via linear interpolation to get the column-wise magnification values of the partially transformed grid, and finally performing the same histogram operation on the columns of the mesh (using the resampled magnification values) to compute the transformed y values. Although this method does produce a transformation which reflects the relative magnification values present in the specification, it encounters problems with non-product magnification distributions. The result of applying this method to the non-product distribution specification of Figure 3.8 is shown in Figure 3.10. Note that the resulting transformation fails to meet our criteria for effective transformations for visualization, in particular the linear areas of magnification do not produce linear regions of transformation, and the transformation is not symmetric about the magnification maxima.

A fundamental problem with the performance of this method is that assuming a

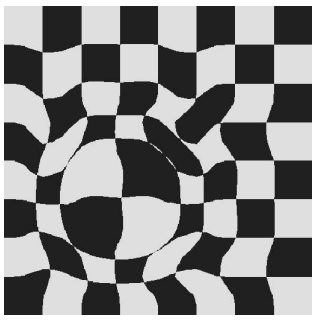


Figure 3.10: Histogram Solution to Specified Magnification in Figure 3.8

rectangular boundary condition on the entire target transformation means that the magnification values will not be interpreted consistently between rows (or columns), since the method of computation is dependent on the sum total of magnification within each individual row and column. Analytical adjustment of the boundary conditions might be able to account for this row and column normalization to some extent, however that would in itself create the additional problem of having non-uniform boundaries on the resulting transformation grid, thus trading one problem for another. In addition there is the problem of ensuring that the local magnification maxima computed in each row and column are properly aligned after the normalization process, and this problem could not be fully addressed by considering boundary conditions alone.

In order to address the shortcomings of these direct methods, we will develop an iterative method which provides a numerical solution to this problem. By dealing with a localized basis for computation, we can simply and directly control the overall

behaviour of the algorithm to produce the desired final result. The general problem is to compute an approximate transformation grid T_C from a specified magnification mesh M_S. The key to our approach is that the ease of converting from transformation to magnification facilitates the conversion in the opposite direction. We compute the implicit magnification M_C from the transformation T_C, and then the magnification error $M_E = M_S - M_C$. We then use M_E to further refine the approximation T_C. On a local node-by-node basis, we are trying to find a transformation t such that:

$$a_t(g(i,j)) = M_S(i,j) \cdot C_a \tag{3.12}$$

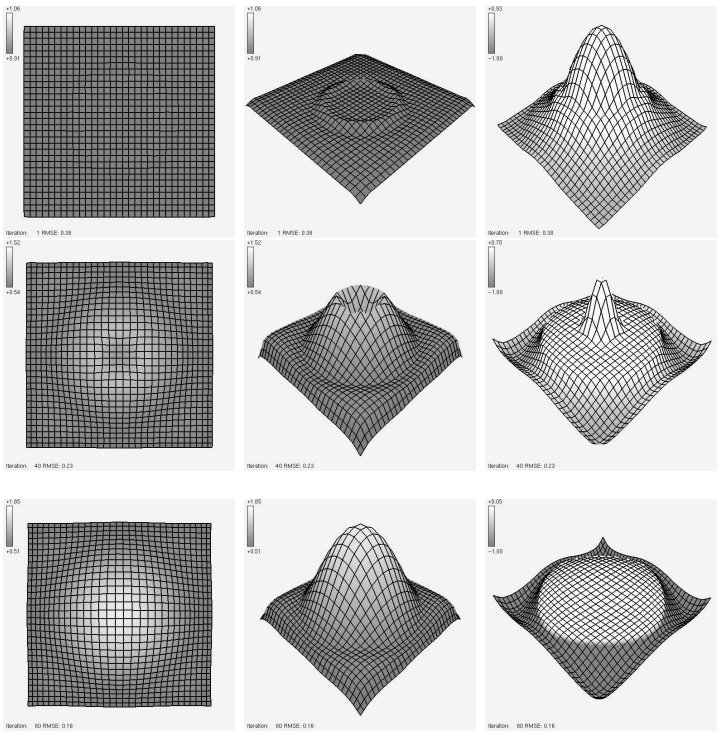
To enhance the visualization of the performance of our method, we join the z magnification values of M_C to the (x,y) coordinates of T_C to create a composite mesh M_{Cv} so that $M_{Cv}(i,j) = (T_C(i,j), M_C(i,j))$. We similarly join the M_E values to T_C giving M_{Ev} with $M_{Ev}(i,j) = (T_C(i,j), M_E(i,j))$. M_{Cv} and M_{Ev} are used for meta-visualization of the performance of the algorithm only, and are not used in any internal calculations.

Conceptually, the algorithm is straightforward. First we initialize T_C to the identity transformation, then for each iteration we compute M_E on a node-by-node basis. If $M_E(i,j) > 0$ then we push the neighbours of (i,j) in T_C away a little bit from $T_C(i,j)$. Conversely if $M_E(i,j) < 0$ then we pull the neighbours of (i,j) in T_C a little

bit closer to $T_C(i,j)$. Both the pushing and pulling operations are easily constrained to preserve the ordering of nodes in T_C (the exact method of constraining is dependent on the specific displacement vector being used, and will be described shortly). The listing below is a simple pseudo-code representation of the basic algorithm. Figure 3.11 shows an example of the operation of this algorithm over a few iterations (using M_S from Figure 3.3); and Figure 3.12 shows how this method handles the multiple, bounded and linear regions of magnification specified in Figure 3.4.

```
// first initialize T_C to identity transform
[ ... ]
// iterate over T_C until convergence
while(!converged)
   for(i := 1..s)
      for(j := 1..t)
         M_E(i,j) := M_S(i,j) - M_C(i,j);
         if(M_E(i,j) > 0)
            pushNeighboursAway(T_C, i, j);
         elsif(M_E(i,j) < 0)
            pullNeighboursClose(T_C, i, j);
         endif
```

As mentioned previously, many different area functions can be used in these methods. The area function used will determine which neighbours should be displaced in our algorithm (*i.e.* if the area function is 4-connected, then the algorithm should



(a) T_C (b) M_{Cv} (c) M_{Ev}

Figure 3.11: Magnification to Transformation Convergence on Iteration 1,40,80

only displace the 4-connected neighbours). Assuming 4-connected displacement, for each $p = T_C(i,j)$, we want to displace each of $\{T_C(i+1,j), T_C(i-1,j), T_C(i,j+1), T_C(i,j-1)\}$. However we will examine the displacement for only one of these neighbours ($q = T_C(i+1,j)$), as the others will be treated similarly. There are two main choices for determining the displacement vector along which to push or pull the neighbours:

1. **Radial:** along the ray defined by p and its neighbour q, or the ray defined by

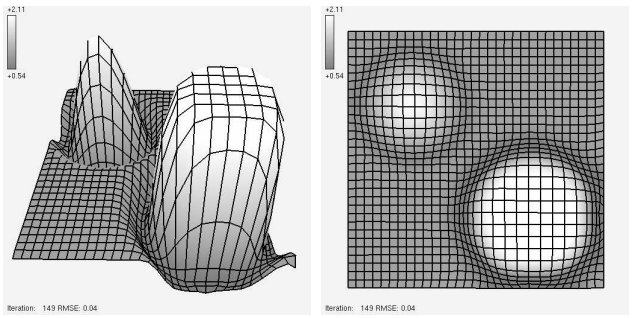


Figure 3.12: M_{Cv} and T_C Computed from Figure 3.4

q and the next neighbour over $r = T_C(i+2, j)$, expressed as[1]:

$$q' = q + (M_E(i,j) > 0 \; ? \; d(r-q) \; : \; d(p-q)) \tag{3.13}$$

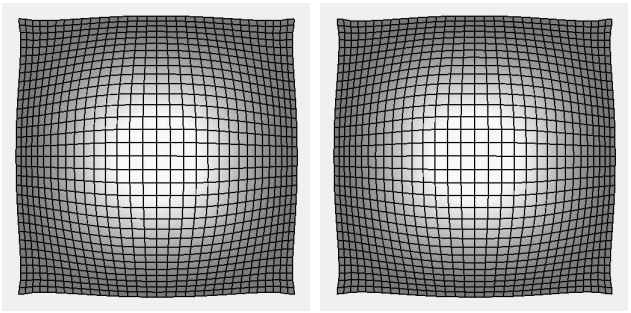
where d is a parametric variable proportional to the absolute value of the error estimate $M_E(i,j)$. By scaling or clamping d to the range $(0 < d < 1)$, we constrain the pushing and pulling to preserve the ordering of the nodes in T_C.

2. **Orthogonal:** In the x or y direction only, depending on if the neighbouring 4-connected node is in an adjacent row or column of the current node. This is equivalent to option 1) above, except that the x or y value is zeroed before incrementing q. For movement in the y direction only, this is can be expressed as:

$$q' = q + (0,1) \cdot (M_E(i,j) > 0 \; ? \; d(r-q) \; : \; d(p-q)) \tag{3.14}$$

[1]Using the ANSI C '?' conditional operator [HS95].

Most of the time the difference between these choices is barely noticeable. The first choice creates a similar effect to the radial transformation described in the previous chapter. The second choice causes the transformation to be slightly similar to the orthogonal transformation from the previous chapter; however the iterative node-by-node nature of the computation does not provide a true orthogonal transformation. Figure 3.13 shows a comparison of these two choices on an identical M_S specification from Figure 3.3. Other M_S specifications may produce somewhat more of a difference between the two methods, but the overall shape of the constructed transformation T_C remains pretty much the same regardless of which method is used. Throughout the rest of this chapter the radial displacement vector type will be used in the examples.



(a) radial (b) orthogonal

Figure 3.13: Comparison of Radial and Orthogonal Displacement Vectors

Going back to our original definition of magnification as a ratio of areas, we recall that this particular definition does not fully preserve all information about

the original transformation. In particular, the single scalar value we use for local magnification does not keep information about those situations where the scaling in the x and y directions is different. In many cases however, the implicit gradient of the magnification field drives the algorithm into a reasonably correct interpretation of the scalar value into two dimensions. As an example of this, Figure 3.14 shows how our algorithm performs on the input magnification from the perspective wall. Note that the algorithm correctly places most of the expansion and compression along the x axis, despite the fact that locally a single value is being used to represent both x and y magnification.

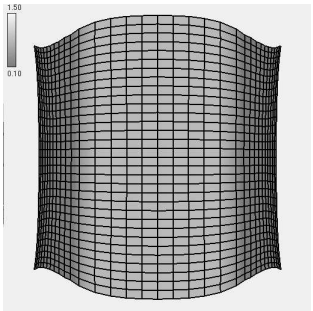


Figure 3.14: Computed Transformation for Perspective Wall

3.3.1 Performance

To measure the convergence of the algorithm, we can use the root mean squared error as the convergence estimate:

$$RMSE = \sqrt{\frac{1}{(p \times q)} \sum_{i=1}^{p} \sum_{j=1}^{q} M_E(i,j)^2} \tag{3.15}$$

The algorithm converges faster if we multiply the error $M_E(i,j)$ by the specified magnification $M_S(i,j)$, so that regions of higher magnification will be more strongly weighted. We also use a refinement coefficient C_r to scale the amount of error that is applied to neighbouring nodes. When $C_r = 0$ no displacement occurs, whereas $C_r = 1$ causes the algorithm to attempt as much displacement as possible on each step of the iteration while still preserving ordering. To an extent, higher values of C_r cause the system to converge faster, but if C_r is too high the approximation will tend to oscillate and possibly not converge at all. The default $C_r = 0.3$ typically provides good results. C_r can also be manipulated as the algorithm progresses; beginning with a high value of C_r and slowly reducing it to 0 as the algorithm converges can provide a type of annealing which may allow for faster overall convergence.

The local error $M_E(i,j)$ can be distributed evenly over the neighbouring nodes; however the algorithm converges faster if we weight the displacements based on the

distances between a node and its neighbours. If $M_E(i,j) > 0$, we weight the displacements so that closer neighbours are pushed a greater distance than farther neighbours. If $M_E(i,j) < 0$, we weight the displacements to pull more distant neighbours a greater distance than closer neighbours. By doing this, we move those nodes which are most responsible for the absolute error the greatest amount. This also serves to balance out the distances between a node and its neighbours, so that the distance for each neighbour approaches the average of all the distances of the neighbours, thus tending towards a uniform scaling in both the x and y directions.

This algorithm converges independently of the complexity of the magnification field, although there a several ways in which complexity can be defined. There is an intuitive visual sense for the complexity of a field which can be obtained through simple visual inspection. As an example, for the magnification fields of Figures 3.3 and 3.4 it is quite clear which of the fields is the more complex. Although this intuitive meaning of complexity is adequate for such clear-cut cases, it is not sufficient for distinguishing between fields which are quite similar, and does not have a quantitative capability. We can compute something similar to the intuitive meaning of field complexity by finding the integral of the gradient of the magnification field. In Section 3.5 we will refer to this gradient field as the "distortion" that is implicit in the magnification field. This integral definition for complexity does not quite capture the true meaning of complexity that we are interested in however, and also does not

take into account the absolute level of magnification specified.

There is another level to the complexity which is of interest however: the complexity of the code and computation required in a traditional foci-based system to form the transformation having that implicit magnification field. With existing foci-based systems for nonlinear magnification (including the one described in the previous chapter) it is quite clear when additional complexity is involved in a transformation. Adding multiple foci usually involves computing multiple transformations for each point and then combining them, or else requires extra attention to the regions between foci. Creating discrete bounded domains for the transformation adds complexity by testing if the point is in the domain and constraining the transformed point to that domain. Making distortion-free areas of linear magnification often requires extra work as well. The timing data of the previous chapter quantify the effect of increasing complexity within that system. As mentioned at the beginning of this chapter, it is this meaning of complexity that we are going to address. Although we do not have a formula for computing this complexity, we can heuristically judge it by noting the number of foci, bounded regions, and linear/nonlinear combinations. The timing data in this chapter will support the statement that convergence time for this algorithm is not dependent on this "foci-based complexity" of the transformation.

A significant determining factor of convergence speed is the volume of specified magnification, or more precisely the volume of error in M_E at iteration 0. For our

discrete magnification meshes we can approximate this by treating M_E as a triangular mesh, the distance between nodes in this mesh is a constant h, and the volume for each triangular cell (assuming that the cell does not intersect the $M_E = 0$ plane, in which case the cell must be re-triangulated) is approximated by (with $\delta \epsilon \{-1, 1\}$):

$$v(i, j, \delta) = \frac{h^2}{2} \frac{|M_E(i, j) + M_E(i + \delta, j) + M_E(i, j + \delta)|}{3} \tag{3.16}$$

and the error volume over the entire mesh is:

$$\mathcal{V}_E = \sum_{i=1}^{p-1} \sum_{j=1}^{q-1} v(i, j, 1) + \sum_{i=2}^{p} \sum_{j=2}^{q} v(i, j, -1) \tag{3.17}$$

Other measures of this "total error" having greater or lesser accuracy are possible, in fact this discrete measure of error is nearly identical to the 1-norm for the error mesh:

$$\mathcal{V}'_E = \sum_{i=1}^{p} \sum_{j=1}^{q} |M_E(i, j)| \tag{3.18}$$

In practice either of these measures seems to be sufficient for characterizing the amount of error that needs to be reduced by the algorithm.

Figure 3.15 shows the effect of altering the overall volume of the error on convergence time for a number of different transformations. For these examples, each

series represents a single type of transformation which is timed with a varying parameter that changes the overall error volume of the transformation. For the series which have 'F' as the second letter, different values of the smoothing filter from 1 (null transformation) to 0 (full transformation) are used. For the series which have 'B' as the second letter, different values of β are used to increase the volume and degree of magnification. A key to the identifier codes is in Table 3.1, and a full listing describing each series, along with an explanation of the series letter codes, can be found in Appendix A.

All timings in this chapter were obtained on a 195 MHz MIPS R10000 CPU. The convergence threshold for *RMSE* is set to 0.05; this value represents the point beyond which further computation does not usually result in visually distinguishable refinements in the transformation. Occasionally lower values of *RMSE* are needed, however this value is reasonable for purposes of comparing timings between series.

clip	vary	combination	source type	constrain
Error	**B**eta	**A**verage	**F**lat/radial	**B**ound
None	**F**ilter	**C**lip	**O**rthogonal	**U**nbound
	Number	**O**verlap	**R**adial	
		Single		

Table 3.1: Key for Timing Series Identifiers. Each series is identified by a 5 letter code, the meaning of each letter position is described here.

Inspection of the results shown in Figure 3.15 reveals a rough correspondence

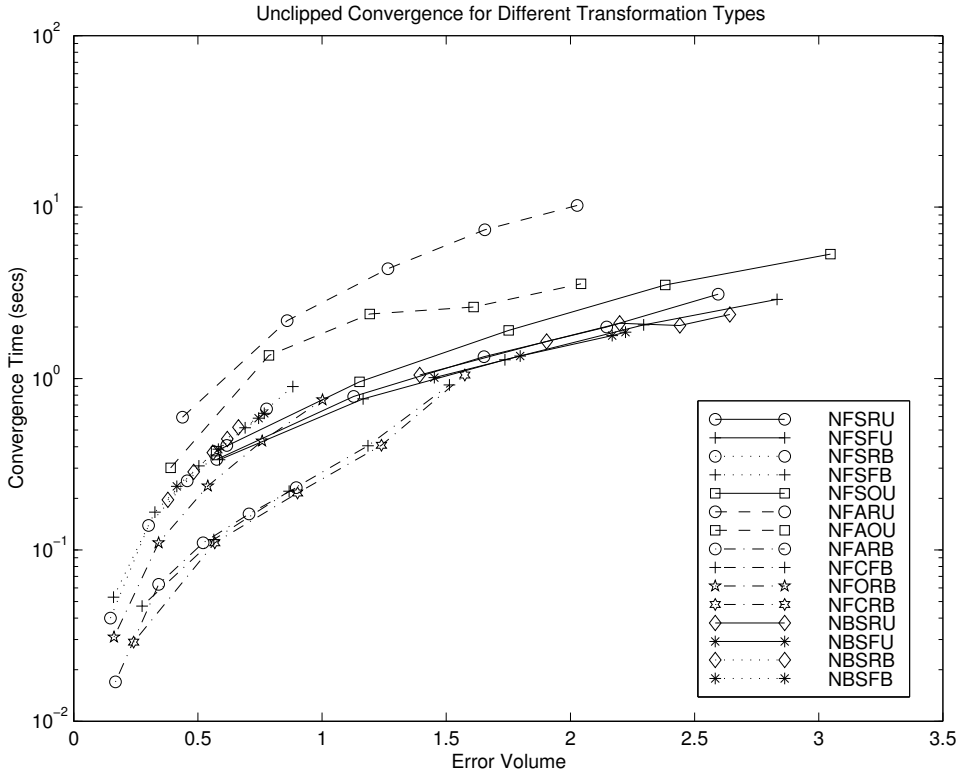
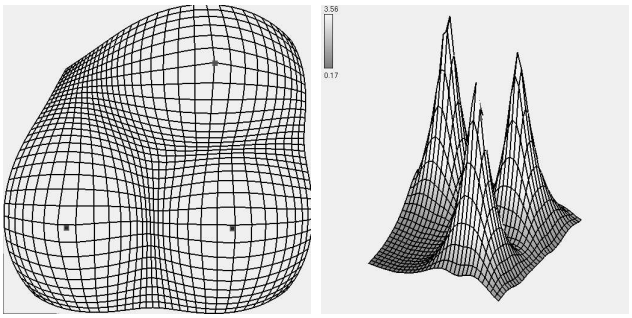


Figure 3.15: Effect of Error Volume ($\mathcal{V}_E$) on Convergence Time

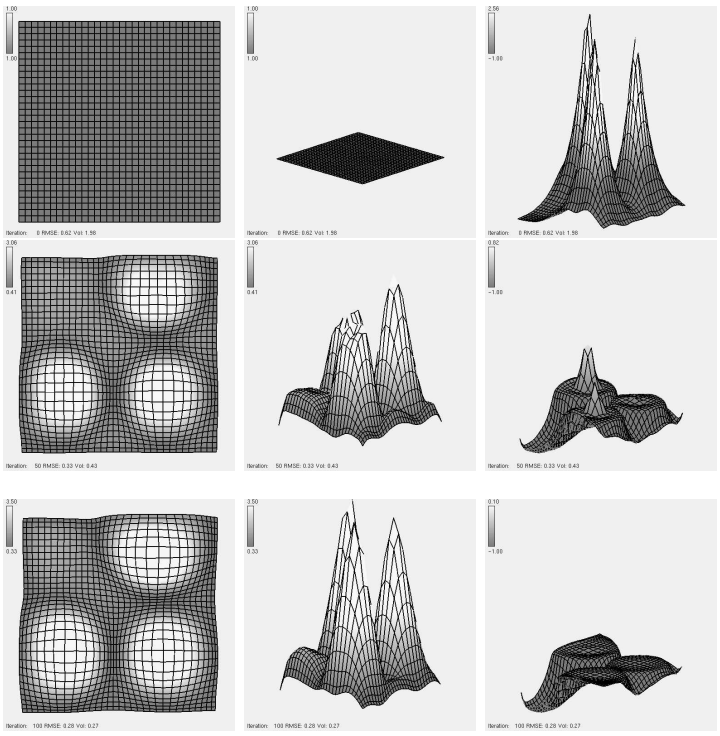
between error volume and convergence time, although clearly there is still a degree of variability in that correspondence. Examination of the outliers on this graph may provide insights into the performance of the iterative algorithm. Series NFARU shows the greatest deviation from the performance trend; this series represents the transformation resulting from averaging several unconstrained radial transformations, as shown in Figure 3.16. This figure shows that the transformation has a large region of specified demagnification (magnification value less than 1), almost the entire upper-left quadrant has a magnification value of less than 0.5.



(a) T_C (b) M_S

Figure 3.16: Outlier Series NFARU Specification

To test the hypothesis that it is this large region of demagnification which is causing the slow convergence, we can examine the performance of the algorithm over a number of iterations, as shown in Figure 3.17. Note that the regions of high magnification converge quickly; the error has been greatly reduced from those regions by iteration 50, and almost completely eliminated by iteration 100. In contrast, the main region of demagnification has barely changed even after 100 iterations of the computation. This is due to the algorithm's weighting of the computation in favour of higher regions of magnification, based on the assumption that is those regions which will be of greatest interest, and which will therefore be most important to compute accurately and quickly. The second highest outlier (NFAOU) is similar to NFARU, except that the orthogonal transformation is used instead of the radial transformation. Both series present similar large regions of demagnification.

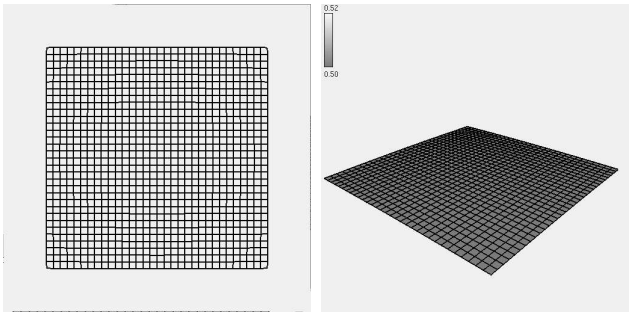


(a) T_C (b) M_{Cv} (c) M_{Ev}

Figure 3.17: Convergence for Series NFARU on Iteration 0,50,100

There is still a possibility that it is not the region of demagnification by itself that is causing the slow convergence, but rather the neighbouring regions of high and low magnification. To test this, we can look at the performance of the algorithm on an input specification which specifies demagnification across the domain with very little variation. An example of this is shown in Figure 3.18. This transformation is identical to the radial transformation in Figure 3.3, except that the β parameter has been decreased to lower the magnification values to between 0.50 and 0.52 (compared

to between 0.53 and 1.95 in Figure 3.3). The algorithm requires 2380 iterations to converge using the input specification from Figure 3.18, compared to 118 iterations when the input specification from Figure 3.3 is used. This further supports the assertion that regions of low magnification cause slow convergence for this algorithm, and that higher regions of magnification will generally tend to converge faster.



(a) T (b) M_S

Figure 3.18: Specification of Uniform Demagnification

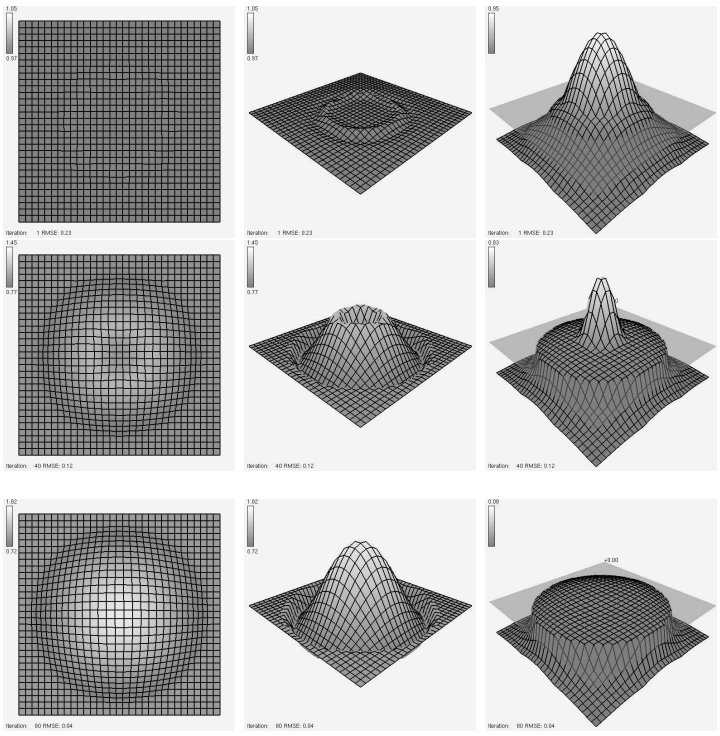
We can explicitly inform the algorithm of what levels of demagnification to ignore using mesh clipping techniques. First we create an error clipping constant C_e with a default value of 0.0, and then have the algorithm ignore nodes where $M_E(i,j) < C_e$. This causes the algorithm to converge faster because it does not have to compress an area whose implicit magnification is greater than its specified magnification. The result of this is that regions of *demagnification* are not strictly enforced, but allowed

to remain at their original unmagnified level (excepting where magnified regions push into those demagnified regions). This allows the resulting transformation grid to fill up the available space more efficiently, sometimes eliminating the empty screen regions which were outside the range of the original transformation grid. Figure 3.19 shows a few iterations on an input identical to that used for Figure 3.11, except that error clipping with $C_e = 0.0$ is now used. When this error clipping method is used, we redefine our error measure as:

$$RMSE = \sqrt{\frac{1}{(p \times q)} \sum_{i=1}^{p} \sum_{j=1}^{q} ((M_E(i,j) > C_e) \; ? \; M_E(i,j)^2 : 0)} \tag{3.19}$$

and observe that the primary determining factor of speed of convergence is now the volume of M_E above the clipping plane defined by C_e. By increasing C_e to 0.1 or 0.25, little visual difference is apparent in the resulting T_C, although substantial performance benefits occur.

Returning our attention to the outlier series NFARU, we can see in Figure 3.20 that when error clipping is applied during the iterative method the algorithm now converges much faster (0.311 vs. 10.231 CPU seconds, 83 vs. 1800 iterations), although the result within the magnified regions is virtually identical. Figure 3.21 show how the performance of the algorithm increases when error clipping ($C_e = 0.0$)

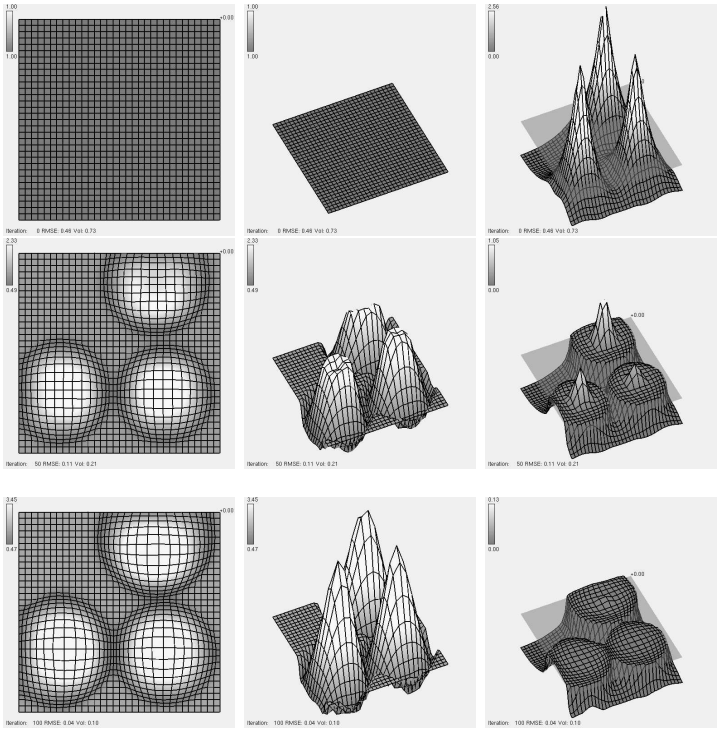


(a) T_C (b) M_{Cv} (c) M_{Ev}

Figure 3.19: Convergence with Error Clipping on Iteration 1,40,80

is used for the examples in Figures 3.19 (NFARU) and 3.20 (NFSRU). The dashed lines represent the performance of the algorithm on those same series when clipping is used. Note that applying the error-clipping decreases the total volume of error, and additionally decreases the computation time required for convergence. Notice also that while NFARU is slower than NFSRU when clipping is not used, the situation is reversed when clipping is used, so that EFARU is now faster that EFSRU. This is clear support for the earlier statement that this algorithm converges independent of the complexity of the magnification field, since EFARU is a much more complex field

specification than EFSRU.



(a) T_C (b) M_{Cv} (c) M_{Ev}

Figure 3.20: Error-Clipped Convergence for Series NFARU (EFARU) on Iteration 0,50,100

In a similar fashion we can define a magnification clipping constant C_m, and make our algorithm ignore nodes having $M_S(i,j) < C_m$. Depending on the particular M_S being used, increasing C_m to 0.5 or 0.75 can significantly increase performance with little cost in the final visual result. Error clipping is more flexible than magnification

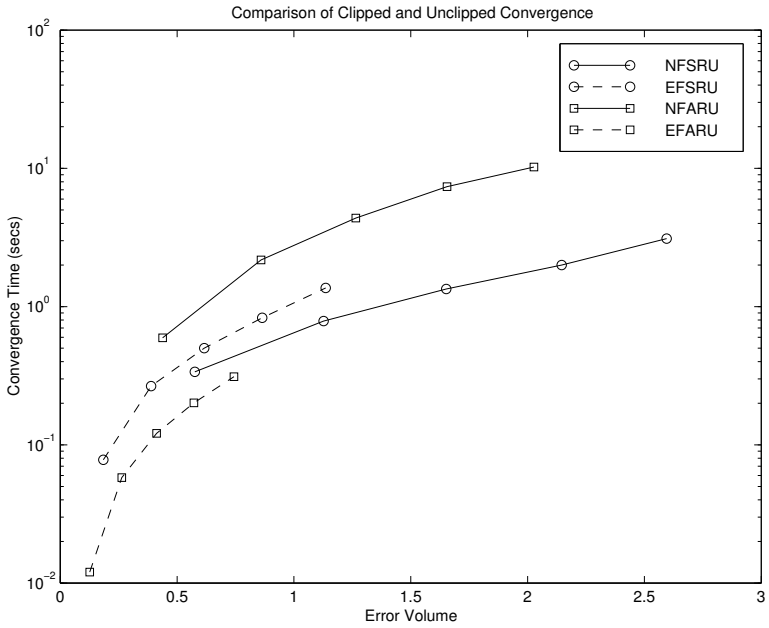


Figure 3.21: Effect of Error Clipping on Convergence Time

clipping however, because it takes into account the changes to the implicit magnification of T_C as the algorithm progresses, and thus distributes the magnification more evenly over the entire domain. By carefully adjusting C_m and C_e for the particular application, very significant increases in performance can be achieved, to the point where our algorithm converges at speeds which are suitable even for interactive applications requiring high frame-rates. Table 3.2 summarizes the effect of adjusting the clipping parameters on a 32×32 mesh with the radial transformation specification from Figure 3.3.

Significant performance increases are also possible through the use of multi-resolution mesh computation techniques. One straightforward approach to realizing this is easily

C_m	C_e	Iterations	Time (s)
-	-	154	0.826
	0.00	72	0.236
	0.25	50	0.104
0.75	-	73	0.220
	0.00	72	0.198
	0.25	50	0.072

Table 3.2: Clipping Performance Using M_S from Figure 3.3

achieved when the mesh contains $(2n+1) \times (2m+1)$ nodes. The iterative computation can then be performed using a stride of length 2 to step through the row and column indices, greatly reducing the number of nodes that need to be computed. After convergence (or a given number of iterations) the remaining nodes are positioned using linear interpolation between the computed nodes, followed by a number of iterations over the full mesh with stride = 1 to fill in any details which may have been missed by earlier iterations. This method can be extended multiple levels to obtain greater benefits, although artifacts of the coarser resolution of computation may become evident unless more sophisticated methods are used to balance the computation between levels. A clear area for further refinement of this algorithm would be to incorporate well-established multi-level computation techniques such as those surveyed in [Dou96] to obtain greater performance. Figure 3.22 shows the effect of changing stride on convergence times for an unconstrained radial transformation while varying the smoothing filter to change the error volume.

Not unexpectedly, mesh resolution is also a significant factor in the performance

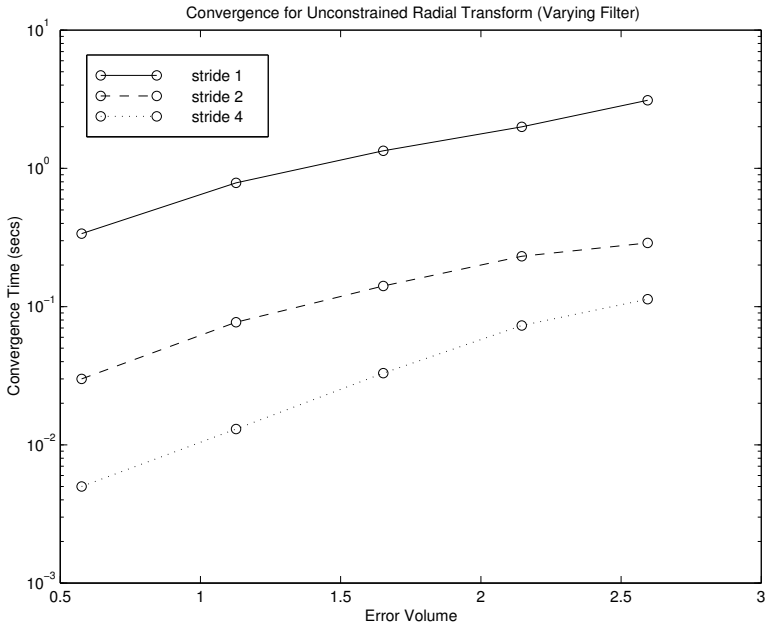


Figure 3.22: Effect of Stride on Convergence Time

of our algorithm. High resolution meshes are able to compute finer detail than lower resolution meshes, but generally require greater computation time. We can parameterize the mesh resolution to tune the performance based on speed/accuracy tradeoffs, the results are shown in 3.23 for the radial transformation with all clipping disabled.

3.3.2 Degenerate Field Specifications

Not all possible magnification specifications will have a solution; there are some degenerate specifications which are physically impossible to satisfy without violating the desired characteristics for nonlinear magnification. Although these physically degenerate cases have no exact solution, our algorithm still manages to compute a

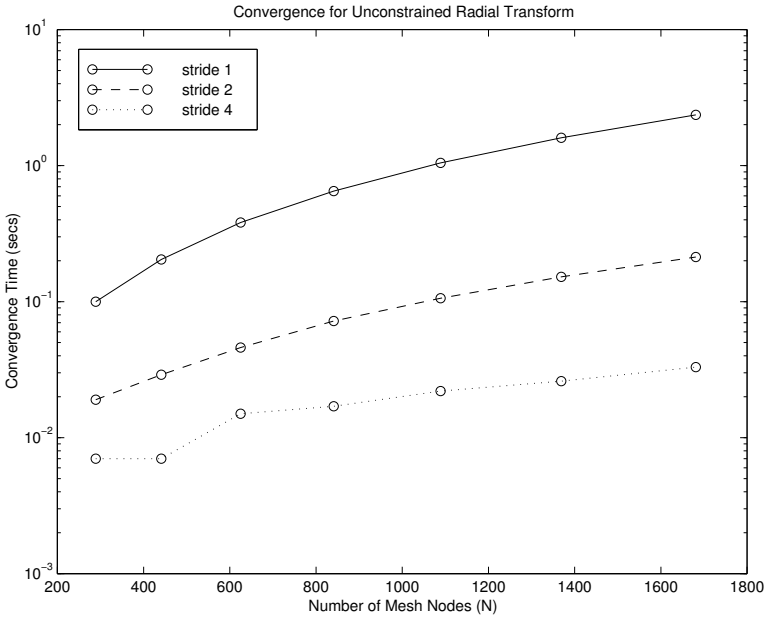


Figure 3.23: Mesh Resolution vs. Convergence Time

reasonable compromise for them. We will examine the degenerate cases below, along with how they can be dealt with effectively. First however it should be noted that any magnification field which is generated from a transformation meeting our stated requirements for nonlinear magnification will by definition never be degenerate, and our iterative method can always manage to construct a transformation grid having the same implicit magnification field as the original transformation grid.

The first type of degenerate case occurs when the specified magnification requires an area greater than the available area. For example a mesh specifying $2\times$ magnification across the entire field cannot possibly be satisfied while maintaining the desired properties of non-occluding in-place magnification. This reflects the intuitive notion

that every expansion should cause a corresponding compression at some other region. Although this notion holds true in general for this algorithm, we do not as yet have an explicit mechanism for enforcing it. In particular, we have not yet introduced any boundary constraints on the transformed image of the domain to ensure that boundary($\mathcal{D}$) = boundary($t(\mathcal{D})$). In Section 3.5.1 we will examine ways in which boundary conditions can be enforced to ensure that $t(\mathcal{D})$ always fits precisely into the region defined by $\mathcal{D}$.

Another related type of degenerate case involves conflicting regions of magnification. For example when a region of very high magnification is surrounded by regions of low magnification (a doughnut shape), it may be physically impossible to satisfy both specifications simultaneously.

We can resolve these degenerate cases by making the assumption that regions of high magnification (and high error) should take higher priority than regions of low magnification. In this way C_e and C_m can always be adjusted so as to relax the specification to a non-degenerate configuration. In addition, weighting M_E by M_S (as described previously) will resolve degeneracies above the clipping planes by emphasizing higher magnification values. It is worth noting that we can also emphasize areas of demagnification simply by reversing the clipping tests and weighting by the inverse of the magnification. We will see how effective these methods can be for resolving degenerate specifications in Section 3.5, where all of the manipulation examples involve

degenerate field specifications. This ability to resolve degenerate cases is a major feature of the satisficing iterative method; direct methods providing exact solutions would have to be modified to produce a satisficing solution when a degenerate field is specified (or the field would have to be constrained to non-degenerate configurations), thus adding much greater complexity to the computation and/or specification task.

While this iterative method is robust in the presence of degenerate specifications, the *RMSE* error estimate does not always converge to 0 when the algorithm has converged to its satisficing solution on a degenerate case. The *RMSE* error estimate does converge along with the algorithm however, so it is possible to monitor the change in *RMSE* between iterations to determine the degree to which the computed solution has stabilized. Although *RMSE* decreases monotonically during convergence in most cases, it very occasionally increases slightly (≈ 0.01) after the algorithm has converged on a satisficing solution; more divergent behaviour has not been observed with this method. This "backspin" can be eliminated by reducing the refinement coefficient C_r as the algorithm converges on the solution. More sophisticated and expensive error estimates might compare the actual transformation grids between iterations on a node-by-node basis to determine the current volatility of the computation; however it is unclear whether the increased computation required for this would result in a significantly more accurate error estimate.

3.4 Combining Magnification Fields and Transformation Grids

We can combine a transformation grid T and its implicit magnification mesh M to enhance the visual perception of the effects of the transformation at a meta-level. We have already seen in the previous section how the $\{x, y\}$ coordinates of T can be joined to the $\{z\}$ coordinate of M to form a composite mesh M_v ($M_v(i,j) = (T(i,j), M(i,j))$). Orthographic projections of M_v from directly above allow the elevation values to define colour gradients which provide effective visual cues to the degree of magnification implicit in a transformation. This technique has already been used to enhance the visualization of T_C in the figures shown in this chapter.

We can also produce a blend of T and M which allows the viewer to visualize the transformation as smoothly shifting into a perspective view of a landscape of magnification values. The underlying mechanism is a simple linear interpolation between T and M. One way this mechanism can be used is based on a viewing model in which the view direction (`lookat`) is aimed at the center of the mesh, while the angle of elevation θ varies from 0 (looking at the mesh edge-on) to $\pi/2$ (looking at the mesh from directly overhead). If we let $t = 2\theta/\pi$, and T_I be an identity transformation, then the $\{x, y\}$ values of M_v are given by $(1-t)T + tT_I$, and the $\{z\}$ values are given by tM so that $M_v(i,j) = ((1-t)T(i,j) + tT_I(i,j), tM(i,j))$. The

effect of this is an aerial view of the transformation grid which can be "pulled down" to view in perspective as a landscape of magnification values. Figure 3.24 shows a few snapshots of this operation.

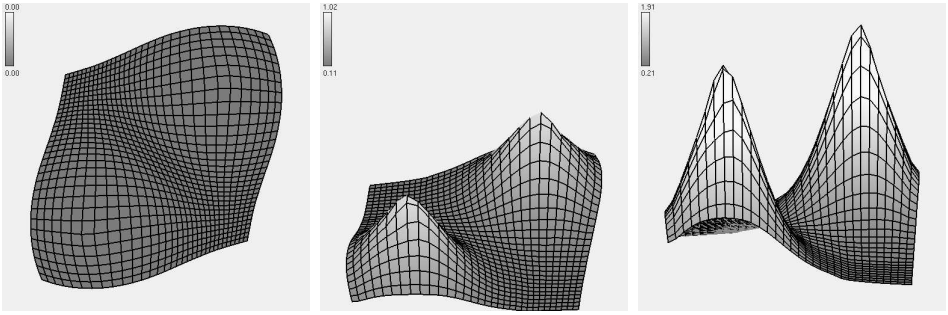


Figure 3.24: Blending a Transformation Grid with its Magnification Field

3.5 Magnification Field Manipulation

By isolating magnification field specification from the transformation function, it is now possible to manipulate the magnification values *directly* rather than have them change only as a side effect of changes in the transformation. We refer to systems which rely on transformation functions to produce nonlinear magnification as *transformation-based* systems. Another class of systems use a physical viewing model and perspective projections to produce nonlinear magnification effects [MRC91, RM93, CCF95a, MK97]. In general such *perspective-based* systems have an irregular correspondence between elevation and magnification, and require careful attention to the orthogonality of surface normals to the view vector, thus adding

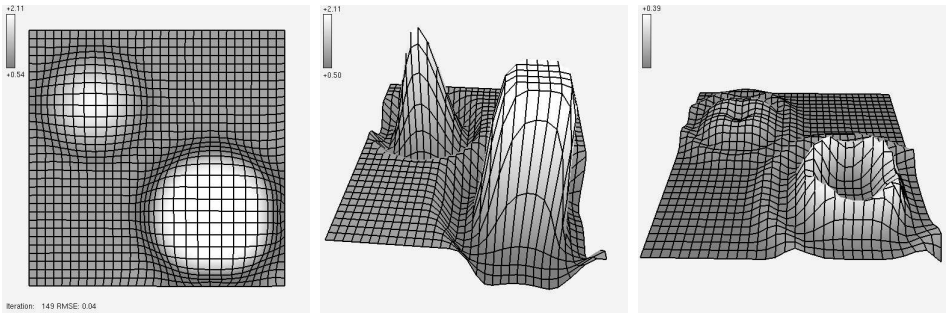
additional degrees of complexity to the magnification specification task. In contrast, the system presented here is a true *magnification-based* system. Direct manipulation of the magnification mesh makes for a system which is both simpler and more expressive than previous nonlinear magnification systems. From the user and application standpoint, the task now is simply to specify desired magnification levels in a scalar field; the conversion techniques in Section 3.3 then construct a transformation grid having those magnification values (assuming the case is not degenerate, and that such a transformation grid is possible). This frees the user and application program from the often difficult task of determining what combination of complexly interacting transformation functions or surface normals will produce the desired magnification.

The remainder of this section will highlight some of the ways in which direct magnification field manipulation can be used, from low-level node operations to high-level global constructions. In all of these examples, we begin with a direct magnification specification and end with a transformation reflecting that specification. The intent is to illustrate through these examples that "magnification" is a more intuitive and useful interface concept than "distortion". Distortion is not the goal of this nonlinear magnification system but only a by-product, although the system does allow for explicit representation of the distortion present in any nonlinear magnification field. Defined as the rate of change in magnification, distortion is easily computed as the slope of the magnification field, as shown in Figure 3.25. For this specific example,

the unsigned slope was computed as:

$$\text{unsigned slope} = \frac{1}{4} \sum_{\Delta i \epsilon \{-1,1\}} \sum_{\Delta j \epsilon \{-1,1\}} |M(i,j) - M(i+\Delta i, j+\Delta j)| \tag{3.20}$$

Other slope calculations such as the magnitude gradient provide similar results. These methods of explicitly computing the distortion of a given transformation point the way for future research into constructing transformations which minimize the distortion. We can imagine an extension of our iterative method which also factors distortion minimization into the local equations in order to find the transformation with the least distortion that satisfies a given magnification specification.



(a) Transformation (b) Magnification (c) Distortion

Figure 3.25: Relationship of Transformation, Magnification, and Distortion

3.5.1 Node-Level Specification

At the lowest level, we can control the magnification mesh on a node by node basis. For demonstration purposes, we have implemented a simple interface which allows the user to select single nodes or rectangular regions of nodes from the magnification mesh. The magnification levels associated with these selected nodes can then be raised or lowered accordingly. This provides a very fine-grained control of the magnification specification. Of greater interest is the ability to associate logical values with the selected nodes. For example the ability to "lock" nodes in place allows for specification of regions which will remain unchanged in the transformation grid. This allows any region of the domain to be excluded from the transformations, as shown in Figure 3.26; these regions can also be locked at magnification levels other than unity by transforming them before locking them in place. In addition, it now becomes a trivial matter to constrain the transformation to any arbitrary bounded domain simply by locking those nodes which define that bounded domain (see Figure 3.26). Some of the examples in this chapter used locked nodes on the mesh perimeter to ensure that the transformed grid would still fit precisely in the original rectangular sampling area.

A major feature of this locking mechanism is that in many cases specifying bounded regions of magnification (or non-magnification) actually *reduces* the computation required (assuming degenerate cases are not introduced). Thus while these bounded regions are similar to the constrained domains introduced in the previous

chapter and the global bounding-box constraints described in [SSTR93], they differ in that additional computation or program complexity is not required here to enforce the fixed boundaries. This locking mechanism allows us to obtain arbitrary bounded and excluded regions for "less than nothing" in computational cost in most cases, by means of a trivial boolean flag check for each node in the iterative conversion process.

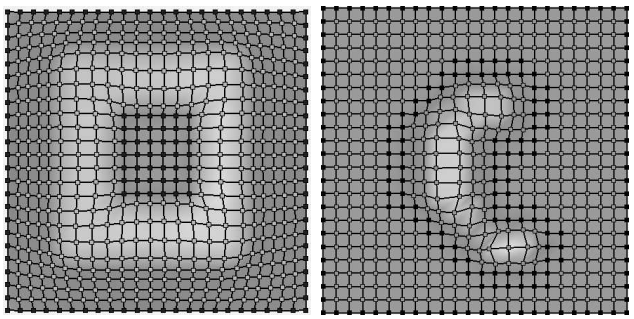


Figure 3.26: Excluded and Bounded Regions of Magnification

More sophisticated constraints are also possible on this node level. For example, we can lock individual coordinates of specific nodes, so that they can only move along a proper subset of the available transformational axes. An example where this is useful is for locking nodes at the rectangular boundary of a domain. Having these nodes completely locked can introduce some extra distortion near the edges; however "floating" edge nodes allow the boundary nodes to float along the boundary edges, so that the overall boundary is still preserved, while allowing some degree of movement of the boundary nodes. This is illustrated in Figure 3.27

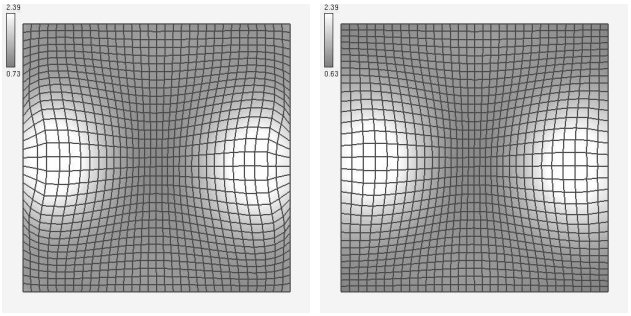
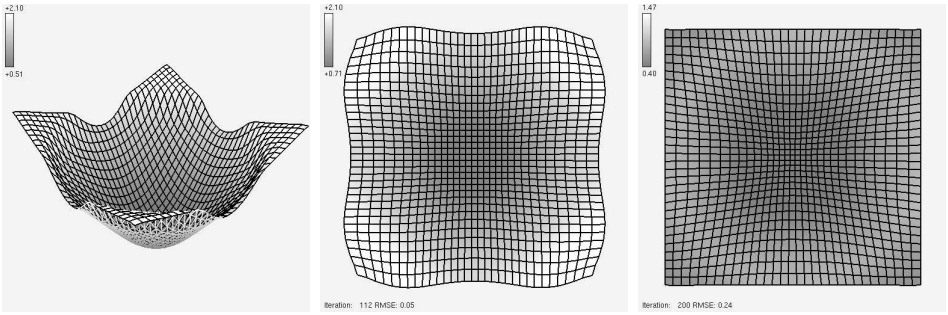


Figure 3.27: Locked and Floating Boundary Nodes

3.5.2 Mesh-Level Operations

Our representation of magnification as a simple scalar field greatly facilitates many operations which would be very involved (if not impossible) with non-magnification-based systems. Given a transformation grid T having an implicit magnification mesh M, it is a simple matter to compute the point-wise inverse mesh M^{-1}, and then find an "inverse" transformation T^{-1} having those inverse magnification values (see Figure 3.28). Further, although our system allows for multiple regions of magnification within a single mesh, it is also possible to combine multiple meshes in useful ways using simple node-by-node operations across the meshes. As examples, two meshes can be blended with proportional averaging: $M(i,j) = dM_a(i,j) + (1-d)M_b(i,j)$ $(0 \leq d \leq 1)$, combined: $M(i,j) = Max\ (M_a(i,j), M_b(i,j))$, or composed: $M(i,j) = M_a(i,j) \times M_b(i,j)$. In addition to operations on the magnification values across the meshes, it is also possible to perform operations on logical mesh values (such as the node-locking mechanism described in the previous subsection). For example we can

find the intersection of the non-locked regions of magnification between two meshes simply by AND-ing their logical values.



(a) M^{-1} (b) T^{-1} (unbounded) (c) T^{-1} (bounded)

Figure 3.28: Inverse of Figure 3.3

3.5.3 User-Level Interfaces

The expressiveness and implementation-independent nature of this magnification representation makes it well suited for the construction of user-interfaces which employ nonlinear magnification. By developing a nonlinear magnification interface as an abstraction layered above our magnification field specification, the designer can construct magnification tools and techniques which are customized to specific tasks.

Perhaps the simplest interface involves construction of a discrete "magnifying glass" which can be moved over the domain; other possibilities are more interesting.

For example, by making the magnification field M_S persistent outside of that same magnifying glass, the user can effectively "paint" arbitrary regions of magnification by stroking the glass (which might now better be described as a brush) over the domain. By using the brush to increment the magnification rather than to set the absolute magnification value, "stroking" a region with the brush would correspond to painting the region with increasing levels of magnification (see Figure 3.29). By using persistence which decays over time (or by not resetting T_C after each movement of the magnifying glass, since T_C will carry some residual implicit magnification from previous iterations), we obtain "trails" of magnification which gradually fade out behind the magnifying glass (see Figure 3.29). This degree of expressiveness goes far beyond anything that can be achieved with existing systems, and moves magnification towards a commodity user-interface item, similar to color and intensity.

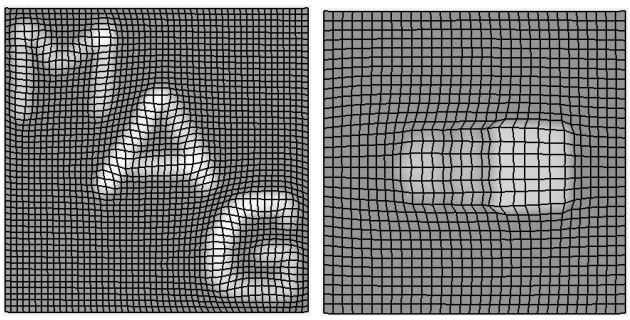


Figure 3.29: Magnifying Brush and Trail

An example of how the magnification brushing technique can be used to good effect is shown in Figure 3.30. By applying texture mapping techniques to the transformation grid, very sophisticated image magnification effects are possible. This leads

to questions about the potential perceptual benefits of subtly enhancing images via these techniques. Many techniques exist for transforming the color space of images through colormap and intensity manipulations to enhance the perception of features of the visual signal. A whole range of image processing techniques (such as edge detection, smoothing, high-frequency filtering) also exist for enhancing or suppressing features of the image. It seems quite possible that nonlinear magnification transformations within the planar space of the image coordinates can be used to similarly enhance perception. An example of a more subtle enhancement is shown in Figure 3.31, which shows how an image from NASA's Mars Pathfinder mission [Nat97] can be enhanced to emphasize the presence and features of the rock referred to as "Barnacle Bill" by the mission specialists. Certainly there is a potential for these magnification techniques to produce misleading visualizations, however the same can also be said for many of the other enhancement techniques. An interesting avenue for further research would be to study the perceptual benefits that might be possible through the use of these subtle image magnifications.

Figure 3.30: Magnification Brushing for Images

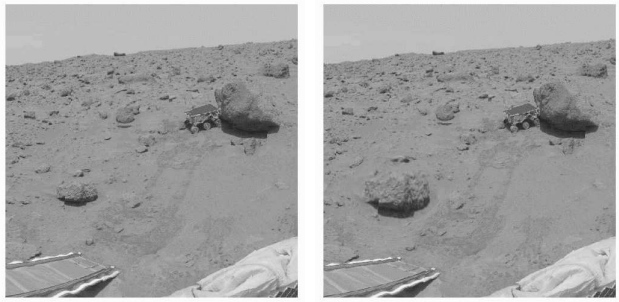


Figure 3.31: Enhancement of "Barnacle Bill". (Original image courtesy of Jet Propulsion Laboratory. Copyright (c) California Institute of Technology, Pasadena, CA. All rights reserved. Based on government-sponsored research under contract NAS7-1260.)

3.5.4 Data-Level Construction

Visualization is but one technique applicable to the exploration of large databases (so-called "data mining"). The greatest potential benefit will combine higher-level (database and semantic-related) mechanisms with low-level (rendering or presentation) ones that are the primary focus of this chapter. The most significant bridge between these levels is to use the data to control presentation, and controlling magnification is a major component of this. One major reason for implementing transformations based on an arbitrary magnification field is to allow properties of the data itself to specify the magnification. When the magnification is entirely directed by human commands, it is only possible to provide a small number of magnification "lenses" which can be easily applied to an image. But much more extensive mapping mechanisms are required when magnification is *data-driven*, since the regions of magnification may potentially have arbitrary shapes.

Using data to indicate regions of special importance is a familiar idea; color coded contour maps display things as concrete as altitude and as intangible as political attitude. For a contour map of environmental pollution, the next step beyond displaying pollution "hot spots" is to expand those regions in order to show the pollution sources within those regions. The exploration of hot-spots for pollution sources can be done by a user-controlled lens, because the situation is static and the task requires only sequential attention to individual hot spots. Automatic magnification becomes truly significant when the information is dynamic or the user's attention must encompass the entire scope at once. An application that displays both of these characteristics is air traffic control. Figure 3.32 shows a simulated air traffic control system where regions of higher traffic density are automatically magnified. As air traffic flows into and out of the region, the number of regions and degree of magnification fluidly changes as the magnification field reflects the movements of planes through the domain.

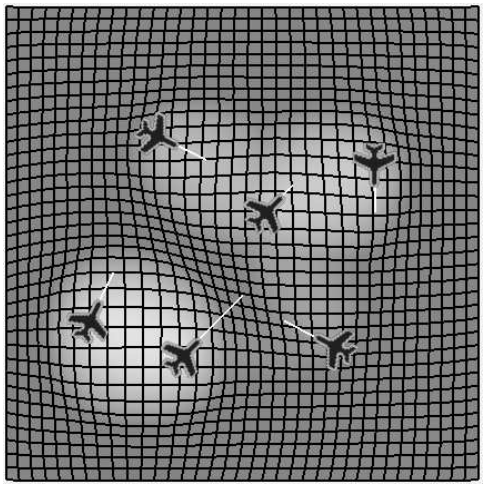


Figure 3.32: Data-Driven Magnification

The air traffic control application is just one example which illustrates the fluid expressiveness possible when using magnification fields for data-driven magnifications. Many other scenarios which use other data properties to define the magnification field are possible. Some examples include: enhancement of image features found during image analysis or machine vision processing (for example, expanding regions of satellite photographs where missile launchers have been detected), expanding the region around tornados or other atmospheric phenomena for weather reports to pinpoint current location within the larger weather pattern, monitoring freeway traffic congestion with magnification of areas of high traffic density or low traffic throughput. Developing applications which are able to use this new functionality to advantage represents a major avenue for future research.

3.6 Related Work

Leung and Apperley [LA94] provide a comprehensive review and taxonomy of major nonlinear magnification systems. Through the introduction of the distinct concepts of transformation and magnification functions, they describe the basic one dimensional properties of nonlinear magnification systems in a systematic fashion. For two dimensions, they use the metaphor of a rubber sheet to describe the behaviour

of nonlinear magnification systems in broad terms. Although useful for conceptualization, this metaphor is not rigorous enough to directly serve as the basis for a constructive theory. One of the goals of this research has been to provide a more rigorous treatment of some of the issues which they raise, in particular the non-trivial magnification to transformation conversion for more than one dimension.

Space-scale diagrams [FB95] are well suited for dealing with typical pan-and-zoom systems; however such systems do not share basic properties of nonlinear magnification systems, such as preserving a view of the global context. The view-dependent nature of space-scale diagrams makes them unsuitable for describing nonlinear magnification systems, as the lines of sight ("great rays") which they use may introduce problems of occlusion for 1D functions having more than one maxima. These problems are compounded further for 2D, and issues of converting magnifications to transformations (and vice-versa) are not addressed in this work.

The pliable surfaces described in 3DPS [CCF95a] may appear to be similar to nonlinear magnification fields, however closer inspection reveals that the two systems have fundamental differences. The most significant difference is that elevation and magnification do *not* correspond directly in 3DPS; the perspective-based nature of 3DPS means that magnification is also dependent on the degree of orthogonality of the surface normal to the viewing vector. The inconsistent relation between elevation

and magnification is readily apparent when considering that with 3DPS both compressed and expanded areas will have a higher elevation than undistorted areas of unit magnification. This points to another difference between the two systems: the way that they produce shading. 3DPS shades regions of distortion using a computationally expensive 3D lighting model, whereas the system presented here shades regions of magnification simply by mapping elevation values into a color/intensity ramp. In addition, 3DPS uses explicit foci to define the magnification, so that increasing complexity of the magnification function entails additional computation. With 3DPS non-occlusion and confinement of data to fixed regions is not inherent, and requires additional (unspecified) constraints on parameters, whereas by its very nature the iterative system described here guarantees non-occlusion and confinement to any size or shape of domain. Also worth noting is that 3DPS is a perspective-based system which is closely tied to its own specific implementation of a physical viewing model. In comparison nonlinear magnification fields are less implementation dependent, and the concepts and techniques can be directly applied to a broad range of visualization and nonlinear magnification systems.

3.7 Coda

Nonlinear magnification fields provide a natural representation for dealing with nonlinear magnification systems. We have seen how the magnification effects of other continuous nonlinear magnification systems can be quantified by constructing implicit magnification fields from their transformations, providing a consistent mapping between complex transformation and magnification functions. Going in the other direction, the iterative method allows construction of a transformation from an arbitrary magnification field specification. This method is simple and effective, even on complex fields having multiple maxima, bounded regions, and areas of linear magnification. It provides reasonable results even when the specified magnification is a degenerate case. A number of parameters can be easily tuned to control overall performance.

The abstract magnification field representation is expressive and easy to manipulate. By removing the restrictions of view dependence and explicit foci, the system provides a natural and intuitive means of specifying magnification which does not rely on the side effects of complexly interacting transformation functions or surface normals. This ease of manipulation can be exploited on a number of levels, from fine-grained control at the individual node level to sophisticated user-interface techniques which can be layered on top of our system. Of particular interest is the ability to use properties of the data itself to define the magnification fields best suited to

visualizing that data, thus opening the door to many new applications of nonlinear magnification.

One area for further research would involve investigating the use of direct methods for magnification to transformation conversion to produce an inital approximation to the computed transformation, and then use a modified version of the iterative method to enforce desired visual properties on a local basis. Another interesting area for further research would be to explore the possibilities of extending the definition of magnification to account for the full mathematical derivatives of the transformation. Although this may re-introduce much of the complexity involved with foci-based nonlinear magnification systems (the elimination of which was a major motivating factor for the research described in this chapter), it may also be that the additional complexity offers potential benefits for additional degrees of control over the resulting transformation. In general a more in-depth mathematical treatment of the issues raised through these methods may yield further insights, the focus of the research described in this chapter is more oriented towards practical issues of programmability and visual effectiveness.

The next chapter will illustrate some of the ways that nonlinear magnification can be used in different applications. For some applications the transformation-based system is best suited, for other applications the magnification-based system may be the better choice. In general, the transformation-based system will be chosen

for "simpler" transformations where computational efficiency is paramount, and the magnification-based system is used where more sophisticated transformations are required. In many cases the nonlinear magnification field representation can be used to effectively bridge the gap between these two systems.

4

Application of Nonlinear Magnification

In this chapter we will examine some of the issues that arise when applying nonlinear magnification to specific tasks. The first section will develop a framework for describing the different levels on which nonlinear magnification can be applied, illustrating each level with specific examples. Then we will discuss some of the problems that come out of attempting to effectively integrate different aspects of visualization within nonlinearly magnified spaces, and present solutions to those problems.

4.1 Levels of Application

We can distinguish between three levels for the application of nonlinear magnification, each reflecting a different degree of proximity between the magnification and the "raw" data. This categorization is not perfect, as there is some ambiguity and overlap between levels, but it is presented here to give some idea of the variety of ways in which nonlinear magnification can be used. In order of increasing proximity to the data, the three levels are *Image Level*, *Render Level* and *Data Level*. We will examine each level in turn, and illustrate the concepts involved with specific examples.

4.1.1 Image Level

At the *image level* we are applying the transformations to objects or scenes which have *already been rendered*. This usually means performing transformations on a raster of discrete pixel values, and there are no discrete objects involved other than the individual pixels.

4.1.1.1 Image Magnification. Two examples of transformations at the image level are magnification of digital images (gif, jpeg, etc.), and direct magnification of the computer screen (the contents of the screen buffer). Given a spatial nonlinear magnification system, it is a relatively straightforward task to use texture-mapping techniques [BN76] to map such images onto the transformed spatial coordinates. As

we magnify the image, we create space between the pixels (and also compress multiple pixels into a single pixel at regions of demagnification). There are a number of standard filtering operations available via texture mapping to fill in this space between magnified pixels. One of the more common methods uses bilinear interpolation to set those pixel values. This creates the effect of a continuous image, although the underlying data are actually only discrete samples of a continuous image.

Texture mapping of this sort has been used extensively for mapping complete document images onto physical surfaces for perspective-based magnification [MRC91, RM93]. Using nonlinear magnification *within* images was first described in [KR96b], and subsequently used in [CCF96b]. Both of these systems use foci-based approaches to magnification; much more sophisticated image magnification effects are possible using the magnification brushing techniques described in Section 3.5.3. Several examples of image magnification have already been shown in earlier chapters (see Figures 1.2 2.9 3.30 3.31). As hardware for acceleration of texture-mapping becomes increasingly common, the applicability of these techniques should become much broader.

4.1.1.2 Multi-Level Image Magnification. Simple image magnification does not provide any new information in the magnified areas, but only larger versions of the existing information. Sometimes we would like to dynamically include new information within the magnified areas of the image. The best example of this involves magnification of road maps: as you magnify a region of a country map, it shows the

state map, then the county map, and finally a city map at the highest magnification. In Section 4.2 we will explore some very effective methods for using nonlinear magnification to integrate multiple images (each representing a different "view" of the information space) into a single multi-level image with smooth transitions between levels.

4.1.2 Render Level

Most of the existing work with nonlinear magnification techniques involves the application of transformations to the coordinates of discrete objects *before* the objects are rendered (the most commonly encountered examples involve some instance of graph visualization). We refer to this method as an application of nonlinear magnification at the *render level*. Typically at this level, the main task of the magnification routines is to place the location of objects within the transformed coordinate space before rendering. In some systems, an approximation of the local degree of magnification is also used to determine the level of detail at which each object is to be rendered.

4.1.2.1 MovieSpace. MovieSpace is a program which allows a user to navigate a 2D space of movie titles and select movies of interest. The mapping of movie titles onto the dimensions of this space might initially be something like: (x axis == movie

title, y axis == director). The individual movies are rendered as isolated points in the space. As the center of magnification draws near to a movie, it is rendered in more detail (the title is shown), and the closest movie is rendered showing even more textual information about the movie. The idea of laying out movie attributes along x and y axes is similar to the Film Finder described in [JS94], however the use of nonlinear magnification here allows for more preservation of global context than is possible using the zoom-sliders introduced in that work. MovieSpace was developed for a Multimedia Databases seminar in the Computer Science Department at Indiana University in Spring 1995, Figure 4.1 shows a snapshot of the program in operation.

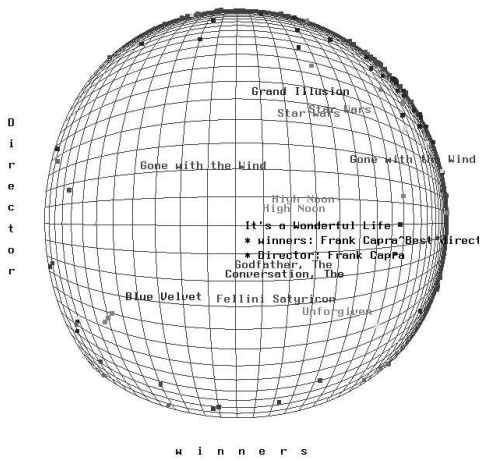


Figure 4.1: MovieSpace

4.1.2.2 Text Browser. Given a plain text file, it is possible to lay out the entire file in a single window, using nonlinear magnification to make some parts of the file

readable while still showing a global view of the file. Figure 4.2 shows one simple implementation of such a text browser, where the spacing between lines of text is adjusted with nonlinear magnification (the size of the text characters themselves remains constant). Full details of this browser are provided in [KM96], along with a user study which provides some support for the hypothesis that this technique might prove useful for viewing highly structured documents such as HTML pages. The intent was to show that preserving a global view of the context would be beneficial for viewing text documents with a high degree of visual structure, such as indentation, headings and whitespace. The hypothesis was not statistically proven by the study however; although users found items faster on average when using the prototype browser, there was a 35% probability that the same increase in performance could be due to random chance alone. The most statistically significant result of this study was that the prototype browser was ineffective when viewing unstructured text documents, and that users found items faster using the normal paging browser for that task. There was also some concern that the slow text rendering routines of the prototype browser which may have skewed the results of the study in favour of the normal paging viewer.

George Furnas originally proposed the idea of inserting and deleting lines from a text file to obtain a type of *generalized fisheye view* in [Fur86]; an idea which

Figure 4.2: Text Browsing

was refined for HTML pages in [BW96a]. More recent work has seen application of nonlinear magnification to the visualization of text files in a groupware environment [GGC96b]. This work is particularly notable in that it provides some controls for directly specifying the 1D magnification function.

4.1.2.3 Graph Visualization – HyperLINK. *HyperLink* [KR96a] is a World Wide Web navigation aid which provides a visual representation of hypermedia traversal as a graph structure. As the user browses through web pages with the Mosaic browser, a graph is constructed in the HyperLink window. The user can click on any node in the graph, and the corresponding web page will be loaded into Mosaic. A nonlinear magnification system is used to enhance spatial separation of nodes near the current node. One center of magnification is always tied to the current graph node, although it is also possible to have multiple centers of magnification. Figure 4.3 shows a snapshot of a sample web browsing session using HyperLink. Similar approaches to

using nonlinear magnification for WWW navigation can be found in [MB95, BHS+96, CDJT96]

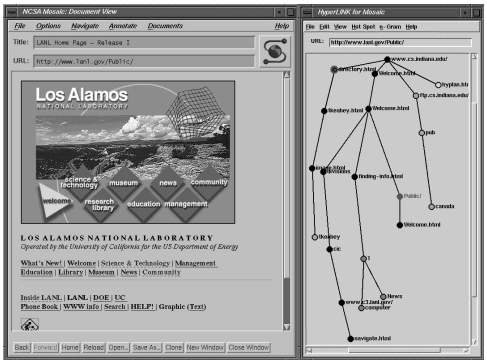


Figure 4.3: WWW Navigation with HyperLink

4.1.2.4 Cluster Visualization for Data Mining. The nonlinear magnification techniques developed in this thesis allow for simple and direct extension to 3 or more dimensions, although visualization in three dimensions inherently involves occlusion of some portions of the data. For some applications this occlusion does not present a great problem; one such example is cluster visualization for data-mining, where the data is composed of dense clusters in a relatively sparse space.

The transformation-based magnification techniques from Chapter 2 have been applied to such a task in a project at Los Alamos National Laboratory, where cluster visualization is being used as part of a data-mining effort for detecting Medicare fraud. Because of the relative sparsity of the data, it was possible to effectively lay out 6

dimensions of the data in a single 3D coordinate system (with $\{x_1, y_1, z_1\}$ rendered in green, and $\{x_2, y_2, z_2\}$ rendered in blue), and then apply constrained nonlinear magnification to selectively expand individual clusters of the data. Figure 4.4 shows normal and magnified views of 6 dimensions of data from this application.

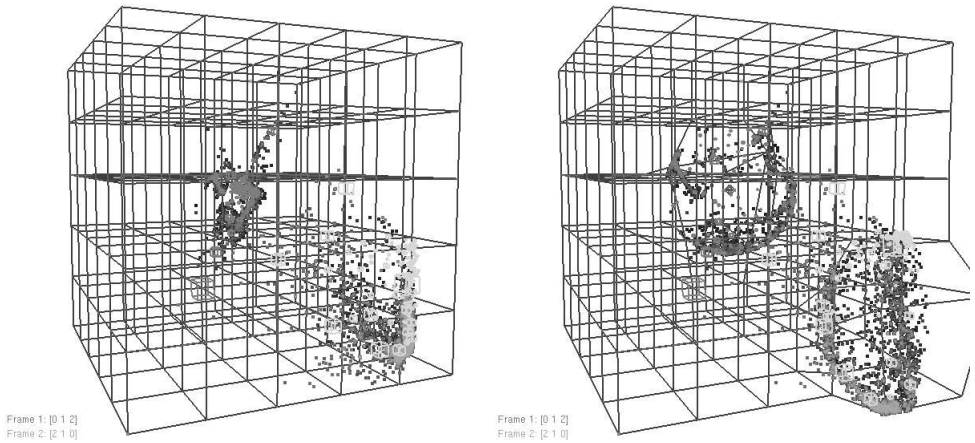


Figure 4.4: Cluster Visualization

4.1.3 Data Level

Data level applications are much more tightly coupled with data access and computation issues than is the case with image and render level applications. There are essentially two ways in which the data and magnification can interact in data level: either the magnification is controlling access to the data, or the data are controlling the magnification.

4.1.3.1 Data Access Based on Magnification Values. Here the magnification values determine how the actual data is *accessed* (both retrieved and arranged). Data level applications might involve interactive extraction of items from a database only when the degree of magnification crosses a certain threshold. Noik [Noi93a] has investigated ways in which fisheye views can open and close subgraph structures within hierarchical graphs.

4.1.3.2 Computation Based on Magnification Values. This technique is similar to data access based on magnification values, except that we are controlling the focus and extent of computation, instead of retrieval from a database. An example of this technique is a wavelet image browser: for rapid visual scanning of large databases of images, it is often not practical to store all of the images in memory simultaneously, and accessing images from disk will introduce problems with latency in data access. We can alleviate this problem by storing compressed wavelet representations of the images, and rendering them at a resolution proportional to the degree of magnification. As the degree of magnification for an image increases or decreases, the inverse or forward fast wavelet transform can be performed on the image incrementally to increase/decrease the size of the image and the amount of high-frequency information available. Figure 4.5 shows an example of this. One problem with this approach involves the discrete $(2^n \times 2^m)$ levels of resolution which are necessary for the pyramidal fast wavelet transform. We will see alternative methods in Section 4.2 where image

resolution levels can be adjusted more smoothly.

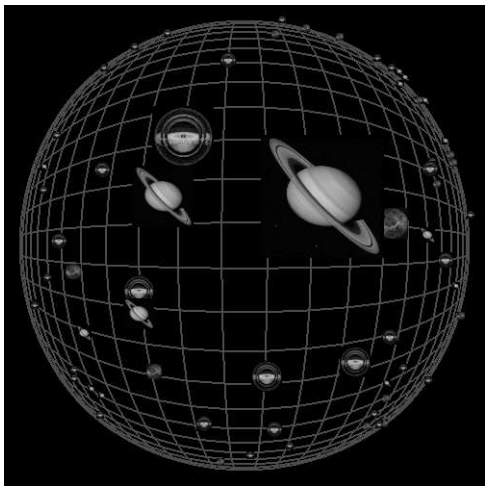


Figure 4.5: Multi-Resolution Image Scanning

There are many other ways in which nonlinear magnification can be used to drive the computation on various problems, beyond the simple wavelets example just shown. For example, a visualization system for observing the simulation of a stress fracture on a plate could typically involve millions of points and take hours to compute and render. We can allow the observing scientist to specify (via specification of nonlinear magnification) which areas of the plate he or she is most interested in, so that the computational and rendering resources can be focussed on obtaining the final result of greatest interest.

4.1.3.3 Data-Driven Magnification. Data-driven magnification offers the unique capability of allowing the data itself to control the magnification transformations. By using properties of the data to directly define the magnification, it is possible to construct transformations which are a "perfect match" to the data. Although some limited degree of data-driven magnification might be possible using more conventional foci-based approaches to nonlinear magnification, this is clearly a case in which the expressive power and ease of manipulation of nonlinear magnification fields can greatly facilitate constructing the mapping between data and magnification. We have already seen several examples of data-driven magnification in Section 3.5.4.

4.2 Putting Detail in Context

In this section we will examine the problem of how to provide suitable detail within magnified regions of a nonlinearly transformed domain. This will not be an exhaustive examination of the issues involved; but rather an exploration of the difficulty of the task, along with some methods which use the tools developed in earlier chapters to provide effective solutions. Throughout the section we will use an "interactive travel atlas" as an application to demonstrate the concepts involved. As regions of the atlas are expanded with nonlinear magnification by a user, points of interest within those regions can be displayed accordingly. For this specific example we will use an atlas of

Scotland, with tourist information such as castles and whisky distilleries as the major points of interest.

4.2.1 The Detail-in-Context Problem

The detail-in-context problem for visualization with nonlinear magnification systems can be stated briefly as:

> How can we effectively utilize the additional space made available by nonlinear magnification transformations to enhance the visualization of the data or objects located in that space?

Figure 4.6 shows a diagram of the problem. Detail can be seen as an additional axis that is orthogonal to the transformational axes.

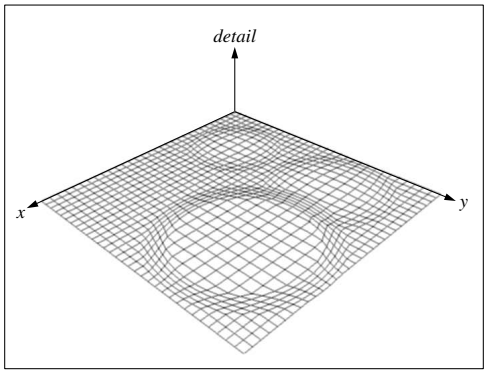


Figure 4.6: The Detail-in-Context Problem

Our formulation of this problem is similar to a number of approaches already taken in the literature. The Perspective Wall [MRC91] was an effort to provide "detail and context smoothly integrated". Researchers at Xerox PARC using hyperbolic display for display of 2D graphs referred to their technique as a "focus+context" technique [LRP95]. Similar terminology is scattered through much of the literature since that time [BW96a, Hol97]. Our definition of the problem has significant differences from these approaches.

Traditional focus+context techniques are designed to create "focus" by enlarging spaces, and reduced "context" by compressing the surrounding spaces. This addresses only half of the detail-in-context problem as it has been defined; it creates the space needed for additional detail, but does not by itself provide a means for placing more detail within that space.

This is not to say that there are no systems which provide enhanced detail within regions of magnification; many such systems exist. The issues that we will address in this section are of a more general nature however: how can we effectively synchronize detail functions with *any* spatial nonlinear magnification transformation? The techniques which we will explore here are designed to be independent of the actual mechanism used to produce the original nonlinear transformation.

4.2.2 Discrete Objects

The first general detail-in-context task that we will examine involves rendering discrete objects within a nonlinearly transformed space. The problem is to determine how to render these discrete objects in a manner that is consistent with the underlying spatial transformation. There are several ways in which this can be approached, ranging from simple object-size calculations to "embedding" objects with the underlying space.

4.2.2.1 Object Size. The simplest method for increasing detail of objects involves only increasing their size. This method is commonly used for single-foci systems such as [SB94], where object size can be based on simple Euclidean proximity to the center of magnification.

However the task becomes more difficult when complex transformations with multiple foci and/or constrained domains are used; simple Euclidean distance is no longer effective as a measure on which to base object size in such cases. A recent article [CCF97] extols the benefits of separating magnification and transformation functions, however the authors do not address the issue of how to ensure that these functions are reasonably synchronized. All of the examples shown in that work involve either a very simple single-focus transformation function or else a very simple magnification

function; for such cases there is no complex interaction of transformation and magnification to account for, and thus simple proximity-based approaches can be used for determining detail. The authors of that work have also redefined the mathematical definition of "magnification function" as originally established in [LA94] into a less quantifiable conceptual "distinction" relating to node-size for graph visualization.

The implicit magnification fields which we developed in Chapter 3 are very well suited for this task when complex transformations are involved. By computing the implicit magnification field for the transformation we can find the magnification for any object within the transformation domain, and render the object with a size proportional to that magnification. This method is general-purpose in nature, and does not require any special knowledge about foci-location or other facts that are internal to the specific transformation technique used to produce the transformation. In addition, since the implicit magnification field is C^0 continuous and well defined over the entire domain it does not leave any gaps where the magnification is undefined, as is the case for some of the other approaches for graph visualization that only define magnification locally at the nodes of the graph. Figure 4.7 shows several examples of how node size can be coupled effectively with implicit magnification values (a similar example is shown if Figure 3.6).

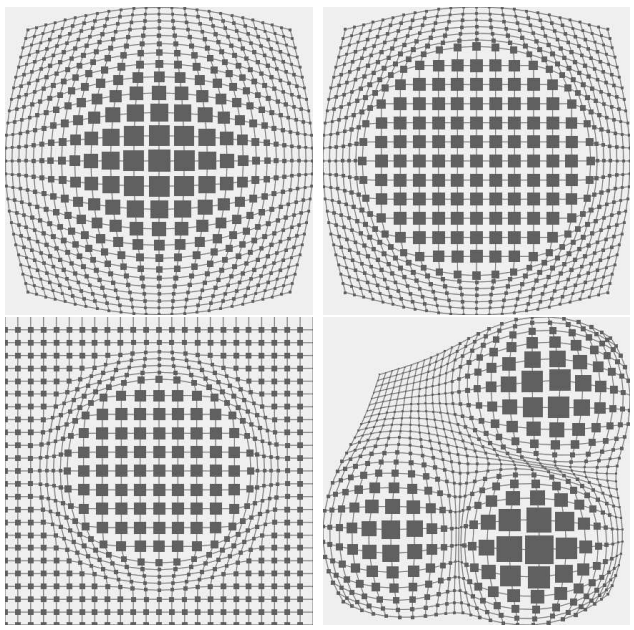


Figure 4.7: Synchronizing Node Size and Implicit Magnification

Figure 4.8 shows how object-size rendering might work for our interactive atlas; each point of interest is rendered as an image which is uniformly magnified proportionally to the implicit magnification of the transformation. Although this example illustrates effective synchronization of detail and transformation functions, it also shows how object size alone is not always sufficient to guarantee that the objects do not overlap each other. In this example we sort the images by magnification level, so that the highest-magnified image will always be completely visible. This sorting can be performed analytically, or on a per-pixel basis using z-buffer rendering. A more sophisticated approach to this problem which uses embedded objects will also be described in Section 4.2.2.3.

Figure 4.8: Interactive Atlas with Variable Object Sizing

4.2.2.2 Level of Detail. We can extend the object-size methods by incorporating *level of detail* (LOD) for the rendering of the objects. Level of detail is a common technique in 3D graphics. It is typically used to suppress details in the polygonal representation of objects when the object is far away from the view-point, since the detail would not be visible at the pixel level anyway.

The LOD notion can be generalized to tasks other than polygonal simplification however, and can incorporate concepts such as semantic levels of detail. Using our interactive atlas example, we might want to represent each castle in the atlas with three levels of detail. At level 0 we can represent the castle by a picture of it, at level 1 we use an iconic representation of a castle (which is shared by all castles), and at level 2 we simply represent the castle by a colored square. Figure 4.9 shows a

schematic representation of these levels of detail.

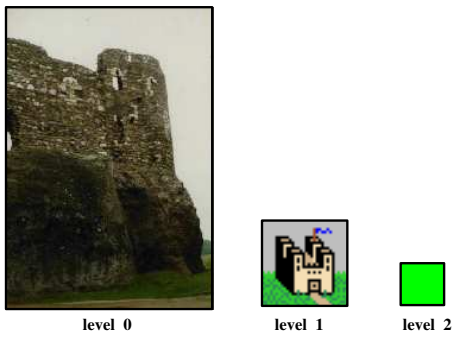


Figure 4.9: Castle Levels of Detail

We can easily drive the LOD rendering of objects simply by making the level of detail proportional to the implicit magnification produced by the transformation. Figure 4.10 shows an example of our interactive atlas using both LOD and object size based on implicit magnification. This is but one simple example of how LOD rendering can be incorporated into nonlinearly magnified spaces, many other approaches are possible; for example the Pad++ [BHP+96] WWW navigation system uses page thumbnails with a LOD function so that the node at the focus of the transformation becomes an actual web page which the user can interact with (follow links, etc).

Using simple linear scaling of objects to synchronize their size with the implicit magnification of the transformation can lead to problems of overlapping objects. In

Figure 4.10: Interactive Atlas with LOD and Variable Object Sizing

addition, this method leads to the perception of the objects as "floating" above the transformed space. Thus while this method binds the dimensions of scale and context effectively, it does not reflect more complex aspects of the transformation itself within the objects.

4.2.2.3 Embedded Objects. An alternative to the simple object-size approach is to *embed* the objects within the transformed coordinate space. This goes beyond simply placing the centers of objects appropriately within the transformed space, and involves mapping the *boundaries* of the objects to the transformed spatial coordinates. Embedding objects in this way produces what could be called a *coherent information space*, where the objects obey the same "transformational physics" as the underlying space. The result is a visualization that has a more tangible aspect to it; the magnification produced by the transformation can now be perceived consistently on three

different levels: the underlying space, between objects, and within individual objects. Prominent examples of this type of embedded object are found in the Perspective Wall [MRC91] and the Document Lens [RM93]. Figure 4.11 shows an example of using embedded objects for our interactive atlas.

Figure 4.11: Interactive Atlas with Embedded Objects

There are a number of problems with embedded objects. The first problem is that magnification level for the objects is *directly* proportional to the implicit magnification, and is therefore not as flexible and responsive as using object size alone, where possibilities such as making object size proportional to the square of the implicit magnification allow for a greater dynamic scaling range for the objects. Another problem with embedded objects is that layout of the objects within the original untransformed

space becomes more of a challenge if we want to ensure that the objects do not overlap at any point along their boundaries, since we need to ensure that the objects do not overlap in the original untransformed space. This difficulty is by no means insurmountable, but does produce limitations on the size and location of objects within the original coordinate space. A final problem with embedded objects is that they may introduce distortion within the objects being magnified. This problem can be dealt with by using transformations with linear regions of magnification to provide uniform scaling within the magnified regions, as shown in Figure 4.12.

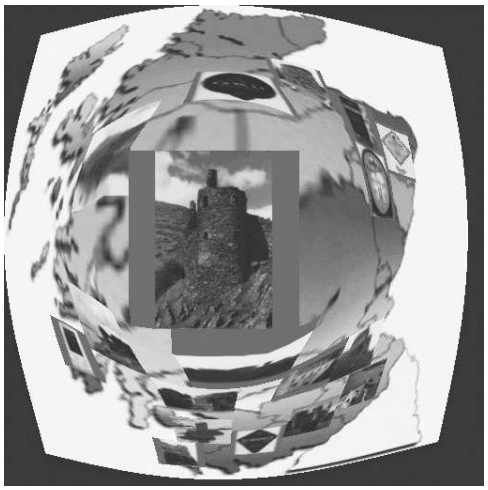


Figure 4.12: Interactive Atlas with Embedded Objects and Linear Magnification

4.2.2.4 Embedded Objects with Size and Level of Detail. We can combine all of the previous techniques within a single visualization system so that we get the advantages of each technique. We linearly scale the objects in the original layout

based on their implicit magnification value in the transformed space (clamping the scale factor so that they do not scale beyond the boundaries defined in the initial layout). Then LOD filtering can be applied, and finally the boundaries of the scaled, filtered object are mapped to the transformed coordinates to embed the object within the transformed space. Figure 4.13 shows an example of this with our interactive atlas. Note that the maximum size of the objects is still bounded with this system, we can remove this restriction by allowing the objects to scale beyond the initial layout boundaries, and using sorting based on magnification (either on a per-object or per-pixel basis) to manage the overlapping objects. This damages the coherency of the information space somewhat, but the extra benefits of having larger objects may make this worthwhile.

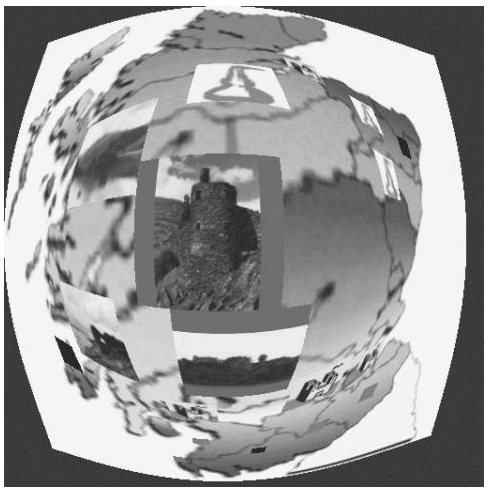


Figure 4.13: Interactive Atlas with Variable Size, LOD and Embedded Objects

4.2.3 Multi-Level Images

Another level on which the detail-in-context problem can be addressed involves integrating different global "views" of the information space. The term "view" in this context does not refer to a different viewpoint, but rather a different visual representation of the information space. In the specific examples described here, each view is represented by a discrete image. This does not represent a fundamental limitation on the types of views or data that can be used, since any data can always be rendered to an offscreen buffer and then used as an image (although this may introduce implementation considerations).

Since we will be dealing with views as images, it is illustrative to look first at the example of nonlinear magnification of a single image. Figure 4.14 shows a simple 8×8 checkerboard image alongside a nonlinearly magnified version of the image. Because the data content is static between normal and magnified versions, no additional information is obtained through the nonlinear magnification, and the only thing that changes is the size (and shape) of the original 8×8 squares. Some filtering techniques which are commonly used in texture mapping may also allow interpolated values between the data squares, however this does not provide any new information to the user, but rather just a smoother transformation between normal and magnified squares.

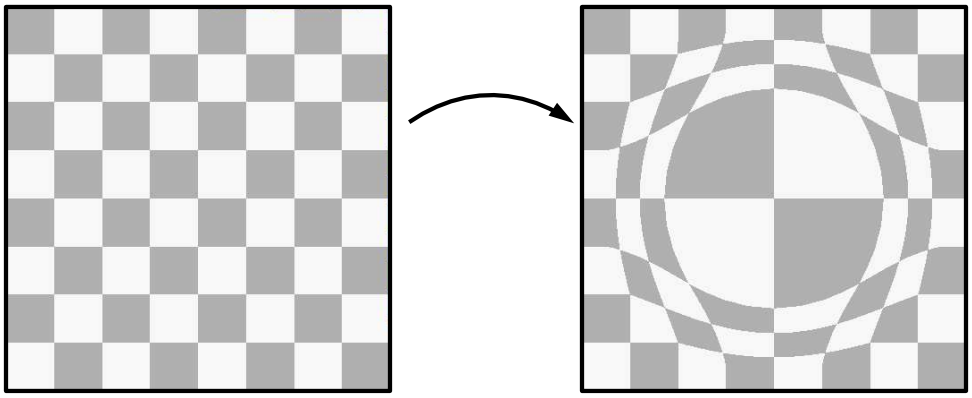


Figure 4.14: Single Image Magnification

We can extend this idea of image magnification to account for multiple levels of images (i.e. multiple views). An example application where this might be useful involves combining state, county and city maps within a single magnified view. At the top level the user is looking at a state map; as the user magnifies some region of the map, the county map is "pulled in" to the magnified area to provide that detail within the state map. Further magnification would also pull in additional detail from the city map.

We will first examine a simple example of multi-level image magnification to illustrate the issues that are involved. Consider two views of an information space, the first view is an 8×8 grid, and the second view is a 16×16 grid. Conceptually, we can think of this process as looking straight-on at the centers of the images, with the 8×8 image in front of the 16×16 image, and filling the entire window. As we apply

nonlinear magnification, we effectively "punch a hole" through the 8×8 grid and pull in the view of the 16×16 grid so that the two views are seamlessly integrated. Figure 4.15 shows an example of this operation. This differs from single image magnification in that we are now dynamically incorporating additional detail within the context provided by our nonlinear magnification transformation.

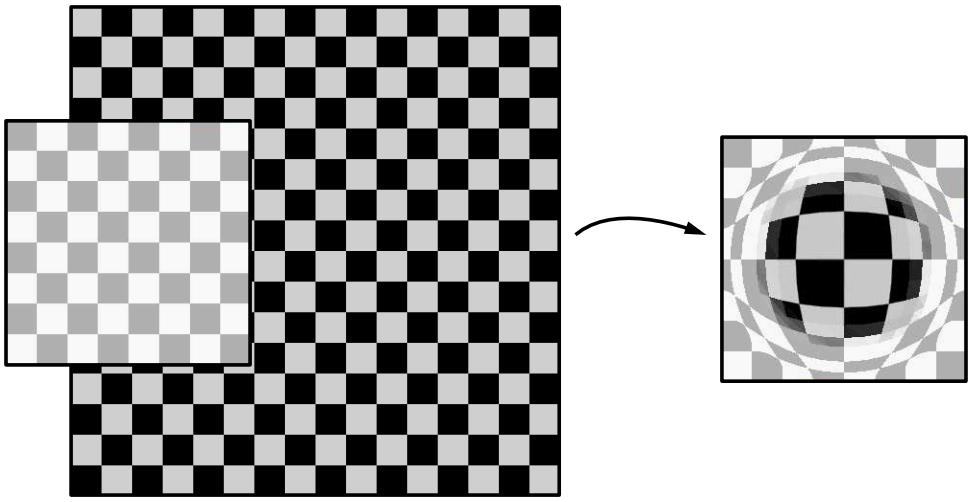


Figure 4.15: Multi-Level Image Magnification

Figure 4.16 shows an example of this using two different maps from the Xerox PARC Map Server [PAR93]. Two views of the California Bay Area are provided, with one map showing more detail (roads, railway tracks etc.) than the other. As the simple view is magnified, additional detail is pulled in from the detailed view. Note that the integration of the two views is seamless, and that all of the map lines are perfectly aligned at the intersection of the simple and detailed views.

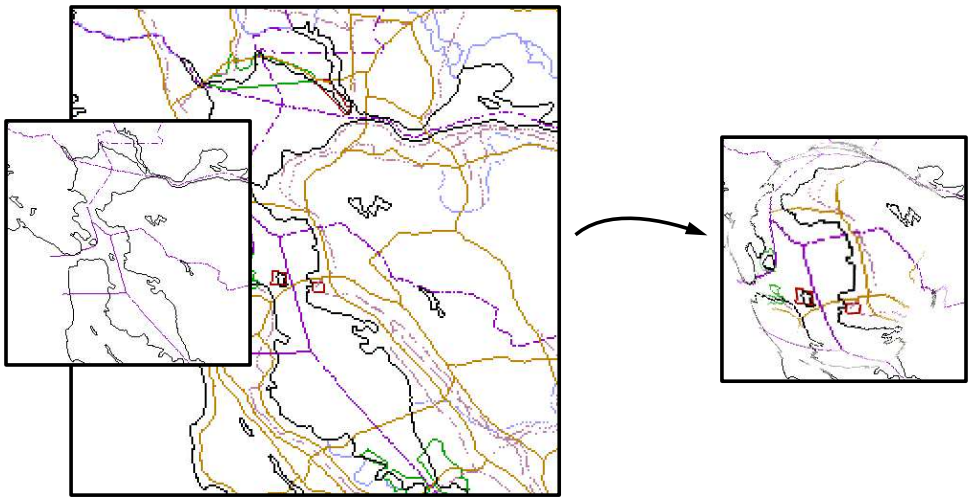


Figure 4.16: Multi-Level Map Magnification

The implementation of multi-level image magnification can be greatly facilitated by the use of MIP-mapping [Wil83], which is a common technique within the graphics community for dealing with texture mapping, and is supported by hardware acceleration on most graphics workstations and PCs that support hardware texture mapping. MIP-mapping is a method for storing different resolution versions of a texture map, so that the most appropriate resolution level can be used for the patch that the texture is being applied to. For example, an $n \times n$ texture will be stored at the original resolution at level 0, along with a filtered $n/2 \times n/2$ version at level 1, and a $n/4 \times n/4$ at level 2 and so on down to a 1×1 version at level $\log_2 n$. Figure 4.17 shows a schematic representation of this for a single channel of an RGB texture.

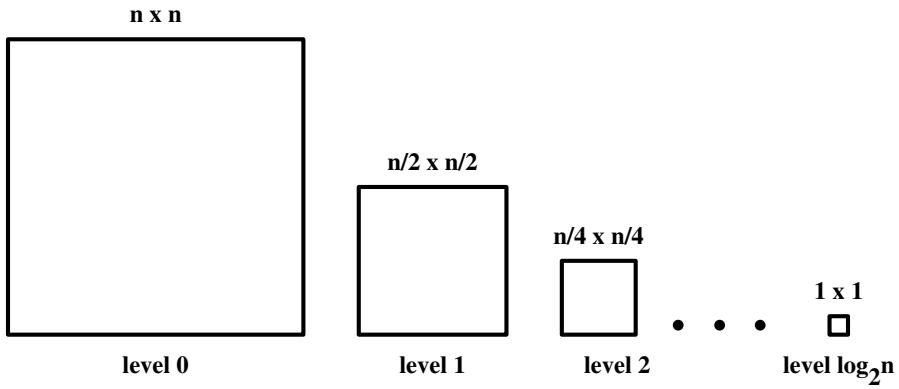


Figure 4.17: MIP-Mapping for a Single Channel

We can bypass the normal filtering construction of MIP-map levels and load any image into the different levels of the MIP-map, as long as the image has the same number of pixel rows and columns as are required for that level. Using our previous example, we can load the 256×256 pixel image of the 8×8 grid into level 1, and the 512×512 pixel image of the 16×16 grid into level 0. If we size our view-port to the same number of pixels as the level 1 image (256×256), we will see only that image; as we magnify portions of the level 1 image, the level 0 image will be pulled into the context of the level 1 image. Although this method works very efficiently on hardware accelerated machines, some hardware implementations also place limits on the size of images that are allowed. This can be a limitation on the maximum size of image (typically 512×512 or 1024×1024), or on the scale factor between levels (most graphics library implementations restrict this to a factor of 2). Although workarounds can usually be found for these constraints, an area for further research

is to develop a more general formalism for describing multi-level image magnification outside of the constrained hardware-accelerated environment, possibly with the goal of setting requirements for a new generation of accelerated texture-mapping hardware which is able to deal with texture levels in a more flexible manner.

4.2.4 Consistent Visual Cues

A final issue involved with the detail-in-context problem is the need to provide consistent visual cues to the user as to what regions are being magnified or demagnified by a given transformation. The implicit magnification fields we developed already provide a quantification for the degree of magnification, our task is simply to render the information in a way that reflects this quantification. We have already seen extensive examples in Chapter 3 where the implicit magnification values are mapped into a 1D color ramp to provide consistent visual cues for a single surface; the situation is somewhat more complicated for textured or multiple surfaces however. One possibility is to use multi-pass rendering and modulate all of the surfaces with the appropriate color ramp values during one of the passes. Another simpler method involves using fog to gradually fade out the regions of lower magnification. This technique is illustrated for a single textured surface in Figure 4.18.

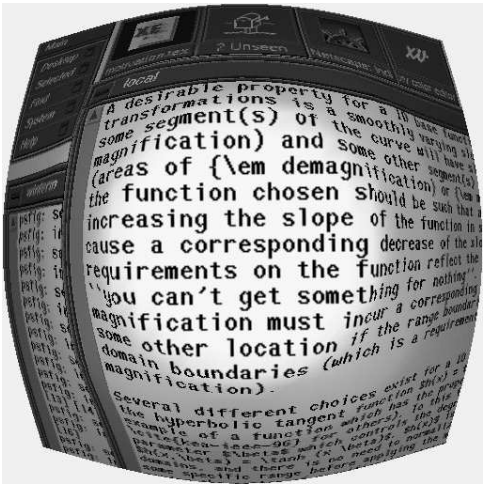


Figure 4.18: Using Fog to Indicate Magnification

4.3 Coda

This chapter has provided a general framework for the application of nonlinear magnification. The framework does not represent a rigid formalism, but rather a simple description of some of the levels on which nonlinear magnification can be applied. Many of these levels are well represented with existing applications, however further exploration of the less populated levels may provide good areas for further research; in particular the possibilities for data-level applications have just begun to be explored.

In this chapter we have also defined the general case for the "detail-in-context" problem, an effort to incorporate specific approaches to the problem within a single implementation-independent framework of general applicability. We also briefly explored some general purpose methods which could be used to address that problem.

5

Related Work

This chapter will provide pointers to the most relevant research on spatial non-linear magnification techniques. Many of the systems mentioned here have already been discussed in greater detail in relevant sections of the preceeding chapters. A good overview of the research up to 1994 can be found in *A Review and Taxonomy of Distortion-Oriented Presentation Techniques*[LA94]. In addition, Noik [Noi94b] has produced a substantial categorization of many systems, especially of those systems specialized for graph visualization. It is worth noting that some of the classification criteria in these overviews do not account very well for the new features of the magnification-based system which we developed in Chapter 3. In particular, the distinction between single focus and multiple foci does not extend well to the direct magnification specification system, where there are no foci in the underlying representation (although user interfaces layered on top of that representation are able to

produce the effect of any number of foci). Rather than reduplicate the material of the earlier overviews, we will instead concentrate on the most relevant systems for spatial nonlinear magnification, drawing on our previous definitions of transformation-based, perspective-based and magnification-based to provide our primary categorization.

5.1 Transformation-Based Systems

The earliest and most commonly encountered systems for producing nonlinear magnification effects are *transformation-based*. These systems use a 2D transformation function of the form

$$(u, v) = t(x, y) \tag{5.1}$$

to directly manipulate the 2D coordinate space. In most cases these methods can be easily extended to higher dimensions ($(u_1, u_2, \ldots, u_n) = t(x_1, x_2, \ldots, x_n)$), although additional issues of occlusion may complicate the visualization task.

Polyfocal Projection [KS78] is a program for static projections of cartographic maps. Although non-interactive in nature, it allowed quite sophisticated transformations, including combining multiple transformations via averaging. Bifocal Display [SA82] provides a central region of high magnification, and 2 neighbouring regions of lower magnification. Multi-Viewpoint Perspective Display [MS91] implements radial

and orthogonal transformations with multiple foci, as well as a "biform" level with two discrete levels of magnification.

Graphical Fisheye Views [SB92b, SB94] provide a simple, efficient transformation function which allows for radial transformations in graph visualization. Nodes are rendered with a size and detail which is proportional to the proximity to the single center of magnification. Hyperbolic Spaces via 4×4 Matrices [PG92] is a novel approach to implementing navigation in hyperbolic space using 4×4 matrix transformations to perform translations and rotations.

Rubber Sheet [SSTR93] is based on an algorithm which uses vector-pair (morphing) techniques to generate transformations. It allows for constrained transformations, areas of linear magnification, and multiple transformations. Pad++ [BH94b] is primarily a pan-and-zoom system, although there is also some application of nonlinear magnification through the use of multi-scale representations of sets of documents. Example applications of this include directory browsing and WWW navigation. Focus + Context [LRP95] techniques for graph visualization in the hyperbolic plane mapped to a Euclidean disk; a distinguishing characteristic of this work is that the edges between nodes are curved to reflect the hyperbolic nature of the underlying space. The largest graph that the authors describe using these techniques with has $1,000$ nodes. 3D Hyperbolic WWW Navigation [MB95] provides a graph visualization of WWW navigation with the graph laid out in 3D hyperbolic space. Similar

to [LRP95], this system also allows curved edges between nodes. This system has been refined and extended in [Mun97], and has been used effectively on hierarchies of 20,000 nodes.

WYSIWIS Groupware [GGC96a] makes use of fisheye views of graph structures. They also describe a "fisheye text viewer" which lays out an entire text document in a window and allows multiple magnification bars (1 per user) to nonlinearly magnify rows of the text. For the text viewer, each user is allowed some control of the degree and shape of the magnification. Magnification of the text is simulated by changing the font size. Distortion Viewing for 3D Data [CCF96a] uses gaussian-sine displacements to build tunnels through 3D data to the region of interest. Although these tunnels are not magnification as we have defined it, they can be used in conjunction with simple unbounded radial and orthogonal 3D transformations. Collaborative GIS Browsers [CPC97] uses a technique based on the techniques in [KR96d] and Chapter 2 to provide fisheye views of GIS systems in a groupware environment.

5.2 Perspective-Based Systems

Another approach for producing nonlinear magnification of 2D surfaces involves mapping those surfaces onto a 3D structure, and then using a perspective projection of that structure to create the effect of nonlinear magnification. For developers of

such systems the magnification specification task involves mapping 2D coordinates of the information space into a 3D coordinate space, along with careful attention to viewing parameters

$$(u, v, w) = p(x, y) + \text{view}() \tag{5.2}$$

Because of this view-dependence and reliance on perspective projections, these systems do not directly scale up to higher dimensions.

Perspective Wall [MRC91] uses graphics hardware to map 2D textures onto a 3D wall with 3 panels. The center panel is perpendicular to the line of sight, and the two adjoining panels slope away from the viewer. Perspective projections of this wall then provide a single area of linear magnification with two areas of reduced resolution to either side. Document Lens [RM93] addresses some of the shortcomings of Perspective Wall, and makes better use of available screen space. 2D information is mapped onto truncated 3D pyramid; the pyramid is the viewed in perspective from above to create the nonlinear magnification effect. 3D Pliable Surfaces [CCF95a] uses perspective projections of suitably constructed meshes to achieve the effect of 2D nonlinear magnification. The mesh is composed of "hills and valleys", with elevation being inconsistently proportional to magnification. In order to avoid occlusion, the hills are slightly tilted towards the view point, although some combinations of

parameters may still cause occlusion. The authors claim computational efficiency as an important design goal for their work, but do not provide any actual performance numbers. Perspective Tunnel [MK97] maps 2D panes of information onto the interior surface of a tunnel with rectangular cross section. Looking down the tunnel with perspective creates the magnification effect. Interesting recursive use of tunnels to create tunnels within the walls of tunnels.

5.3 Magnification-Based Systems

Magnification-based systems represent the most direct method available for specification of magnification. The direct scalar representation creates no multi-dimensional dependencies:

$$z = m(x, y) \tag{5.3}$$

The work described in Chapter 3 represents the only fully magnification-based system to date for spatial nonlinear magnification. Other systems allowing simple manipulation of 1D magnification functions are described in [GGC96a, KR96d].

5.4 Comparison

The following table summarizes the capabilities of some of the most relevant systems for spatial nonlinear magnification. Data is not available to complete all entries with certainty. Those entries for which no information is available are marked with a "?".

	Transformation Capabilities					
	Bound	Linear	Foci	Window	Manipulate	Speed
Transformation-Based						
Polyfocal [KS78]	·	·	○	·	·	NA
Multi-viewpoint [MS91]	·	○	○	·	·	?
Graphical Fisheye [SB94]	·	·	·	○	·	150
Rubber Sheet [SSTR93]	○	●	○	○	○	500
Focus+Context [LRP95]	·	·	·	○	·	1,000
3D Hyperbolic [MB95]	·	·	·	○	·	20,000
Text Viewer [GGC96a]	·	○	●	○	●	?
Thesis (pipeline)	●	●	●	●	●	50,000
Perspective-Based						
Perspective Wall [MRC91]	·	★	·	○	·	?
Document Lens [RM93]	·	★	·	●	·	?
3DPS [CCF95a]	○	●	○	○	○	?
Magnification-Based						
Thesis (field)	★	★	★	★	★	1,500

Table 5.1: Features of Spatial Nonlinear Magnification Systems. (· = poor, ○ = fair, ● = good, ★ = best

Explanation of table categories: *Bound* = constrained domains. *Linear* = combined linear and nonlinear magnification. *Foci* = multiple (or no) centers of magnification. *Window* = deals with window boundaries. *Manipulate* = allows direct

manipulation of transformation and/or magnification (beyond simple parameter adjustment). *Speed* = number of points that can be directly *transformed and rendered* at interactive rates. An interactive frame rate is assumed to be 10 FPS in this thesis, platform descriptions are given in the relevant chapters, along with more precise timings of the isolated transformation routines (without rendering). Refer to the references given for details of other systems.

6

Conclusions

This dissertation introduced the term *nonlinear magnification* to describe the effects common to a wide variety of related systems, and provided a conceptual basis, constructive formalisms, and effective implementations for encompassing all of these effects and more within a single well-defined framework.

We have developed a system for categorizing all of the existing spatial nonlinear magnification systems in terms of their underlying fundamental principles. We have also effectively bridged the gap between transformation and magnification; providing theory and tools for converting between the two. This result allows us to quantify the magnification effects of a wide variety of nonlinear magnification systems, reducing nonlinear magnification from a poorly defined visual or implementational phenomenon to a well defined scalar field of magnification values. This scalar field can then be used to quantify other aspects of the magnification; for example, we are now able to

explicitly compute the distortion present in a given transformation.

In terms of implementation, we have developed two new systems for producing nonlinear magnification. Our first method uses a RISC-type approach to building a *transformation pipeline* composed of sequences of simple and modular 2D transformations. This system is able to produce complex transformations, and is also very efficient computationally, transforming points at rates which are orders of magnitude faster than published numbers for existing systems. The expressiveness and efficiency of this system is greatly facilitated by the use of piecewise linear functions, which allow entire sections of the transformation pipeline to be collapsed in simple table look-up operations.

Our second method introduced the *nonlinear magnification field* as a lower-level representation for nonlinear magnification based on our mathematical formalism of the relation between transformation and magnification functions. We showed how the implicit magnification field of a transformation can be used to quantify the effects of that transformation, and also provided an iterative method for constructing a transformation from a specified magnification field. The scalar magnification field representation is particularly easy to manipulate, as there are no explicit foci or multi-dimensional dependencies, so that direct and fluid specification of magnification is now possible. We illustrated how this ease of expression could be exploited effectively; most notable among these examples was the introduction of magnification brushing

and data-driven magnification.

We also developed a general framework detailing a number of levels on which nonlinear magnification can be applied, ranging from image magnification to data-access tasks. We examined the issue of how to deal with the generalized *detail-in-context* problem, so that we can effectively synchronize detail rendering functions to take advantage of the extra space produced by nonlinear magnification transformations.

These contributions all taken together constitute our framework for nonlinear magnification. It is expected that this framework will help bring structure and common understanding to research in the field, and will continue to serve as a solid foundation on which to build further developments.

A

Description of Transformation Series for Timings

This appendix shows the transformation grids and implicit magnification meshes which were used as the input specifications for the timings of the iterative method for computing transformation grids described in Chapter 3. Each transformation is represented by a 5 letter code which can be interpreted to find the initial specification, as well as a number of parameters used during the timing run. The description of each letter in the code is described below. The first letter indicates whether or not clipping was used in the timing, and the second letter indicates which parameter of the transformation is being varied within the series. The remaining letters identify the transformation type which is used as the specified input magnification. In the figures which follow, the initial two letters are removed from the code, since each transformation type may appear in more than one series with different parameters varying for the series.

```
E - error clipping (errClip = 0.0)
```

```
N - no clipping

.B - varying beta (degree of magnification) (2.0 -> 4.0 step 0.4)
.F - varying smoothing filter (0.0 -> 1.0 step 0.2)
.N - varying N (number of mesh nodes)

..A - averaged transform combo
..C - clipped transform combo
..O - overlap (composition) combo
..S - single transform

...F - flat and radial combined
...O - orthogonal
...R - radial

....B - bound
....U - unbound
```

These series are not intended to provide an exhaustive categorization of all possible inputs, rather they are meant to provide a reasonably representative sampling of the types of input specifications that are likely to be encountered, and which account for the visual functionality of most of the other existing systems for nonlinear magnification.

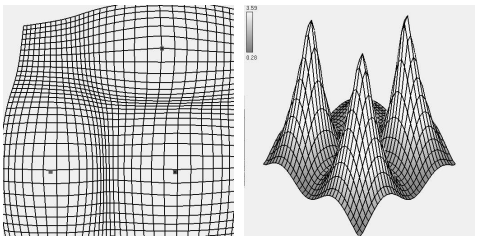


Figure A.1: AOU

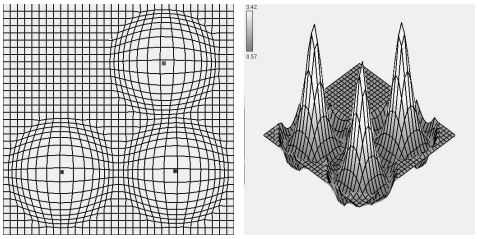


Figure A.2: ARB

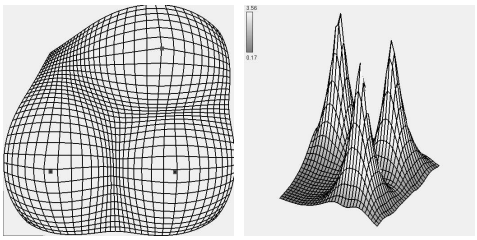


Figure A.3: ARU

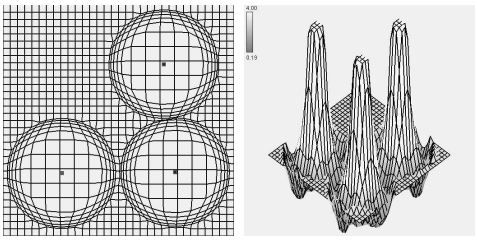


Figure A.4: CFB

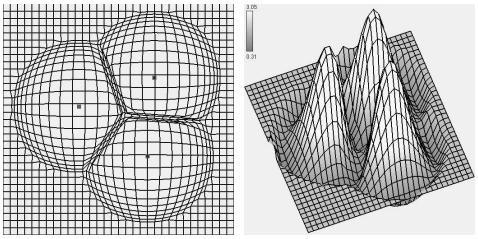


Figure A.5: CRB

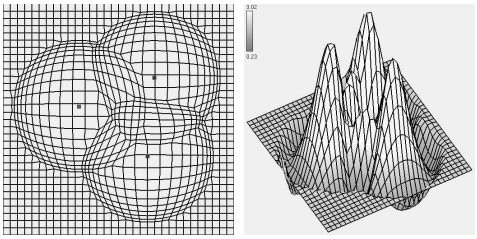


Figure A.6: ORB

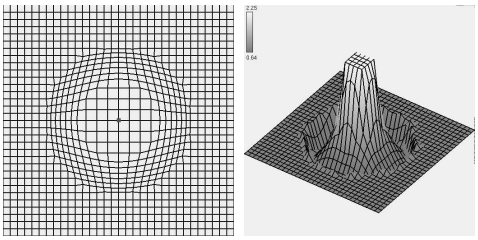


Figure A.7: SFB

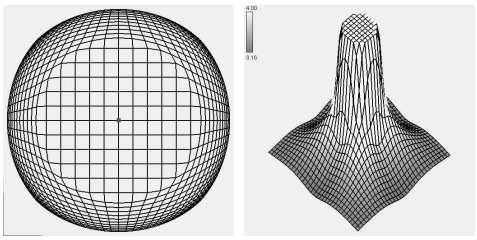


Figure A.8: SFU

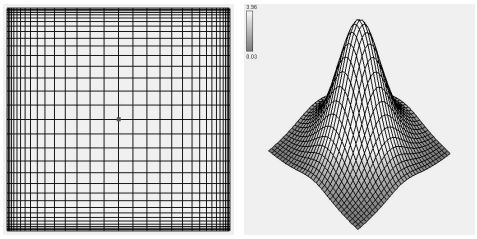


Figure A.9: SOU

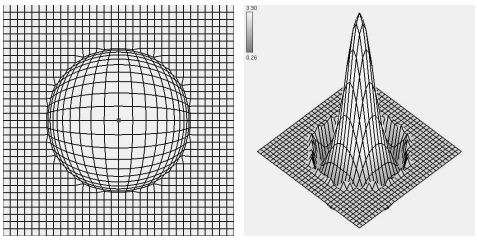


Figure A.10: SRB

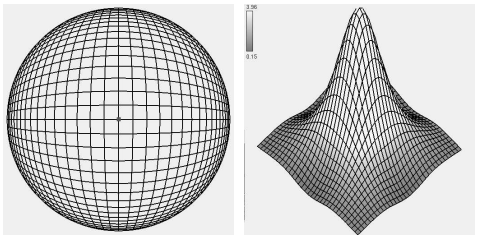


Figure A.11: SRU

Bibliography

[BH94a] Benjamin B. Bederson and James D. Hollan, *Pad++: A zoomable graphical interface*, Proceedings of the ACM Conference on Computer Human Interaction, 1994, Short paper.

[BH94b] Benjamin B. Bederson and James D. Hollan, *Pad++: A zooming graphical interface for exploring alternate interface physics*, Proceedings of the ACM Symposium on User Interface Software and Technology, 1994.

[BHDH95] Lyn Bartram, Albert Ho, John Dill, and Frank Henigman, *The continuous zoom: A constrained fisheye technique for viewing and navigating large information spaces*, Proceedings of the ACM Symposium on User Interface Software and Technology, November 1995.

[BHP+96] Benjamin B. Bederson, James D. Hollan, Ken Perlin, Jonathan Meyer,

David Bacon, and George Furnas, *Pad++: A zoomable graphical sketchpad for exploring alternate interface physics*, Journal of Visual Languages and Computing (1996), 3–31.

[BHS+96] Benjamin B. Bederson, James D. Hollan, Jason Stewart, David Rogers, Allison Druin, and David Vick, *A zooming web browser*, Proceedings of SPIE Conference on Multimedia Computing and Networking, 1996.

[BMNW97a] Marc H. Brown, Hannes Marais, Mac A. Najork, and William E. Weihl, *Focus+context display of web pages: Implementation alternatives*, Tech. Report 1997-010, Digital Systems Research Center, May 1997.

[BMNW97b] Marc H. Brown, Hannes Marais, Mac A. Najork, and William E. Weihl, *Focus+context display of web pages: Implementation alternatives*, Proceedings of the 6th International World Wide Web Conference, April 1997, Also appeared as DEC-SRC Technical Note 1997-010.

[BN76] James F. Blinn and Martin E. Newell, *Texture and reflection in computer generated images*, Communications of the ACM **19** (1976), no. 10, 542–546.

[BW96a] Marc H. Brown and William E. Weihl, *Zippers: A focus+context display of web pages*, World Conference of the Web Society, 1996, Also available as DEC-SRC Tech Report 140.

[BW96b] Marc H. Brown and William E. Weihl, *Zippers: A focus+context display of web pages*, Tech. Report 140, Digital Systems Research Center, May 1996.

[CCF95a] M.S.T. Carpendale, D. Cowperthwaite, and D. Fracchia, *3D pliable surfaces: For the effective presentation of visual information*, Proceedings of the ACM Symposium on User Interface Software and Technology, 1995, pp. 217–226.

[CCF95b] D. Cowperthwaite, M.S.T. Carpendale, and D. Fracchia, *Distortion viewing techniques for 3-dimensional data*, Tech. Report TR 95-06, School of Computing Science, Simon Fraser University, December 1995, Draft of InfoVis '96 paper.

[CCF96a] M.S.T. Carpendale, D.J. Cowperthwaite, and F.D. Fracchia, *Distortion viewing techniques for 3D data*, Proceedings of the IEEE Symposium on Information Visualization, IEEE Visualization, 1996, pp. 46–53.

[CCF96b] M.S.T. Carpendale, D.J. Cowperthwaite, and F.D. Fracchia, *Multi-scale viewing*, SIGGRAPH Visual Proceedings, August 1996, Technical Sketch., p. 149.

[CCF96c] D. Cowperthwaite, M.S.T. Carpendale, and D. Fracchia, *Visual access for 3D data*, Proceedings of the ACM Conference on Computer Human

Interaction, 1996, Short paper.

[CCF97] M.S.T. Carpendale, D.J. Cowperthwaite, and F.D. Fracchia, *Extending distortion viewing from 2D to 3D*, Computer Graphics and Applications **17** (1997), no. 4, 42–51.

[CCFS95] M.S.T. Carpendale, D. Cowperthwaite, D. Fracchia, and Thomas Shermer, *Graph folding: Extending detail and context viewing into a tool for subgraph comparisons*, DIMACS Workshop on Graph Drawing, 1995.

[CCFS97] M.S.T. Carpendale, David J. Cowperthwaite, F. David Fracchia, and Margaret-Anne D. Storey, *Exploring distinct aspects of the distortion viewing paradigm*, Tech. Report TR 97-08, School of Computing Science, Simon Fraser University, September 1997.

[CDJT96] Gerald Collaud, John Dill, Chris V. Jones, and Paul Tan, *The continuously zoomed web - a graphical navigation aid for www*, Proceedings of IEEE Visualization, October 1996, Late breaking hot topic.

[CFC+96] M.S.T. Carpendale, Andrew Fall, David J. Cowperthwaite, Joseph Fall, and F. David Fracchia, *Case study: Visual access for landscape event based temporal data*, Proceedings of IEEE Visualization, October 1996, Short Paper.

[Chu95] Neville I. Churcher, *Photi — a fisheye view of bubbles*, Information and Software Technology **37** (1995), no. 1, 31–37.

[CPC97] Neville Churcher, Parames Prachuabmoh, and Clare Churcher, *Visualization techniques for collaborative GIS browsers*, International Conference on GeoComputation, August 1997.

[DBHH94] John Dill, Lyn Bartram, Albert Ho, and Frank Henigman, *A continuously variable zoom for navigating large hierarchical networks*, Conference on Systems, Man and Cybernetics, 1994, pp. 386–390.

[Dou96] Craig C. Douglas, *Multigrid methods in science and engineering*, IEEE Computational Science and Engineering **3** (1996), no. 4, 55–68.

[FB95] George W. Furnas and Benjamin B. Bederson, *Space-scale diagrams: Understanding multiscale interfaces*, Proceedings of the ACM Conference on Computer Human Interaction, 1995.

[FK95] Arno Formella and Jorg Keller, *Generalized fisheye views of graphs*, DIMACS Workshop on Graph Drawing, 1995, pp. 242–253.

[FPF88] K.M. Fairchild, S.E. Poltrock, and G.W. Furnas, *Semnet: Three-dimensional graphic representations of large knowledge bases*, Cognitive Science and its Applications for Human-Computer Interaction (Raymonde Guindon, ed.), Lawrence Erlbaum Associates, 1988.

[Fur81] George W. Furnas, *The fisheye view: a new look at structured files*, Tech. report, Bell Labs, 1981.

[Fur86] George W. Furnas, *Generalized fisheye views*, Human Factors in Computing Systems, CHI '86 (1986), 16–23.

[FvDFH90] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes, *Computer graphics: Principles and practice*, Addison-Wesley, 1990.

[GG95] S. Greenberg and C. Gutwin, *Sharing fisheye views in relaxed WYSIWIS groupware applications*, Tech. Report 95/577/29, Department of Computer Science, University of Calgary, November 1995.

[GG97] Carl Gutwin and Saul Greenberg, *Interactive fisheye views for groupware*, Tech. report, Department of Computer Science, University of Calgary, 1997.

[GGC96a] Saul Greenberg, Carl Gutwin, and Andy Cockburn, *Awareness through fisheye views in relaxed WYSIWIS groupware*, Proceedings of Graphics Interface, May 1996.

[GGC96b] Saul Greenberg, Carl Gutwin, and Andy Cockburn, *Using distortion-oriented displays to support workspace awareness*, People and Computers XI (Proceedings of the HCI'96), August 1996.

[GW93] Rafael C. Gonzalez and Richard E. Woods, *Digital image processing*, Addison-Wesley, 1993.

[HGD95] Volkmar Hovestadt, Oliver Gramberg, and Oliver Duessen, *Hyperbolic user interfaces for computer aided architectural design*, Proceedings of the ACM Conference on Computer Human Interaction, 1995, Short paper.

[Hol97] Lars Erik Holmquist, *Focus+context visualization with flip zooming and the zoom browser*, Proceedings of the ACM Conference on Computer Human Interaction, March 1997, Poster.

[HS95] Samuel P. Harbison and Guy L. Steele, *C, a reference manual*, 4th ed., Prentice Hall, 1995.

[JS91] Brian Johnson and Ben Shneiderman, *Treemaps: A space-filling approach to the visualization of hierarchical information structures*, Proceedings of IEEE Visualization, 1991.

[JS94] Ninad Jog and Ben Shneiderman, *Starfield information visualization with interactive smooth zooming*, Tech. Report CS-TR-3286, University of Maryland, HCIL, May 1994.

[Kea97] T. Alan Keahey, *the Nonlinear Magnification Home Page*, A WWW resource devoted to all aspects of nonlinear magnification available at:

http://www.cs.indiana.edu/hyplan/tkeahey/research/nlm/nlm.html, January 1997.

[KM96] T. Alan Keahey and Julianne Marley, *Viewing text with non-linear magnification: An experimental study*, Tech. Report 459, Department of Computer Science, Indiana University, April 1996.

[Koi95] Hideki Koike, *Fractal views: A fractal-based method for controlling information display*, ACM Transactions on Information Systems **13** (1995), no. 3, 305–323.

[KR96a] T. Alan Keahey and Reid Rivenburgh, *HyperLINK: A program for visualizing traversal of the WWW*, Proceedings of the ACM Conference on Hypertext, March 1996, Demonstration.

[KR96b] T. Alan Keahey and Edward L. Robertson, *Non-linear image magnification*, Tech. Report 460, Department of Computer Science, Indiana University, April 1996.

[KR96c] T. Alan Keahey and Edward L. Robertson, *Techniques for non-linear magnification transformations*, Tech. Report 455, Department of Computer Science, Indiana University, March 1996, Draft submission of InfoVis'96 paper.

[KR96d] T. Alan Keahey and Edward L. Robertson, *Techniques for non-linear magnification transformations*, Proceedings of the IEEE Symposium on Information Visualization, IEEE Visualization, October 1996, pp. 38–45.

[KR97a] T. Alan Keahey and Edward L. Robertson, *Nonlinear magnification fields*, Tech. Report 478, Department of Computer Science, Indiana University, March 1997, Draft submission of InfoVis'97 paper.

[KR97b] T. Alan Keahey and Edward L. Robertson, *Nonlinear magnification fields*, Proceedings of the IEEE Symposium on Information Visualization, IEEE Visualization, October 1997.

[KRB94] Karlis Kaugars, Juris Reinfelds, and Alvis Brazma, *A simple algorithm for drawing large graphs on small screens*, DIMACS Workshop on Graph Drawing, no. 489, 1994, pp. 278–282.

[KS78] Naftali Kadmon and Eli Shlomi, *A polyfocal projection for statistical surfaces*, The Cartographic Journal **15** (1978), no. 1, 36–41.

[KY93] Hideki Koike and Hirotaka Yoshihara, *Fractal approaches for visualizing huge hierarchies*, IEEE Symposium on Visual Languages, 1993, pp. 55–60.

[LA94] Y.K. Leung and M.D. Apperley, *A review and taxonomy of distortion-oriented presentation techniques*, ACM Transactions on Computer-Human Interaction **1** (1994), no. 2, 126–160.

[LR94] John Lamping and Ramana Rao, *Laying out and visualizing large trees using a hyperbolic space*, Proceedings of the ACM Symposium on User Interface Software and Technology, November 1994, Technical Note, pp. 13–14.

[LRP95] John Lamping, Ramana Rao, and Peter Pirolli, *A focus+context technique based on hyperbolic geometry for visualizing large hierarchies*, Proceedings of the ACM Conference on Computer Human Interaction, 1995.

[MB95] Tamara Munzner and Paul Burchard, *Visualizing the structure of the World Wide Web in 3D hyperbolic space*, VRML Conference Proceedings, 1995.

[MG93] Deborah Mitta and David Gunning, *Simplifying graphics-based data: applying the fisheye lens viewing strategy*, Behaviour and Information Technology **12** (1993), no. 1, 1–16.

[Mit90] Deborah A. Mitta, *A fisheye presentation strategy: Aircraft maintenance data*, Proceedings of INTERACT, 1990.

[MK97] Kenneth Mitchell and Jessie Kennedy, *The perspective tunnel: An inside view on smoothly integrating detail and context*, Eurographics Workshop on Visualization in Scientific Computing, April 1997.

[MPBH95] Jonathan Mayer, Ken Perlin, Benjamin B. Bederson, and James D. Hollan, *Two document visualization techniques for zoomable graphical interfaces*, Short paper submitted to CHI '95., 1995.

[MRC91] J.D. Mackinlay, G.G. Robertson, and S.K. Card, *The perspective wall: Detail and context smoothly integrated*, Proceedings of the ACM Conference on Computer Human Interaction, 1991, pp. 173–179.

[MS91] Kazuo Misue and Kozo Sugiyama, *Multi-viewpoint perspective display methods: Formulation and application to compound graphs*, Human Aspects in Computing: Design and Use of Interactive Systems and Information Management, Elsevier Science Publishers, 1991, pp. 834 – 838.

[Mun97] Tamara Munzner, *H3: Laying out large directed graphs in 3D hyperbolic space*, Proceedings of the IEEE Symposium on Information Visualization, IEEE Visualization, October 1997.

[Nat97] National Aeronautics and Space Administration, *Mars pathfinder homepage*, WWW Page., 1997, http://mpfwww.jpl.nasa.gov/default.html.

[Noi93a] Emanuel G. Noik, *Exploring large hyperdocuments: Fisheye views of nested networks*, Proceedings of the ACM Conference on Hypertext, November 1993, pp. 192–205.

[Noi93b] Emanuel G. Noik, *Layout-independent fisheye views of nested graphs*, IEEE Symposium on Visual Languages, August 1993, pp. 336–341.

[Noi94a] Emanuel G. Noik, *Encoding presentation emphasis algorithms for graphs*, DIMACS Workshop on Graph Drawing, no. 489, 1994, pp. 428–435.

[Noi94b] Emanuel G. Noik, *A space of presentation emphasis techniques for visualizing graphs*, Proceedings of Graphics Interface, May 1994, pp. 225–234.

[Noi96] Emanuel G. Noik, *Dynamic fisheye views: Combining dynamic queries with mapping and database views*, Ph.D. thesis, Department of Computer Science, University of Toronto, 1996.

[PAR93] Xerox PARC, *Map viewer*, 1993, www.parc.xerox.com/istl/projects/mapdocs/.

[PG92] Mark Phillips and Charles Gunn, *Visualizing hyperbolic space: Unusual uses of 4x4 matrices*, Proceedings of the 1992 Symposium on Interactive 3D Graphics, ACM SIGGRAPH, March 1992.

[PM64] Murray H. Protter and Charles B. Morrey, *Modern mathematical analysis*, Addison-Wesley, 1964.

[PRS97] Bernhard Preim, Andreas Raab, and Thomas Strothotte, *Coherent zooming of illustrations with 3D-graphics and text*, Proceedings of Graphics Interface, May 1997, pp. 105–113.

[RM93] G. Robertson and J. D. Mackinlay, *The document lens*, Proceedings of the ACM Symposium on User Interface Software and Technology, 1993, pp. 101–108.

[RMC91] G.G. Robertson, J.D. Mackinlay, and S.K. Card, *Cone trees: Animated 3d visualizations of hierarchical information*, Proceedings of the ACM Conference on Computer Human Interaction, 1991, pp. 189–194.

[SA82] Robert Spence and Mark Apperley, *Data-base navigation: An office environment for the professional*, Behaviour and Information Technology **1** (1982), no. 1, 43–54.

[SB92a] Manojit Sarkar and Marc H. Brown, *Graphical fisheye views of graphs*, Proceedings of the ACM Conference on Computer Human Interaction, May 1992.

[SB92b] Manojit Sarkar and Marc H. Brown, *Graphical fisheye views of graphs*, Tech. Report 84, Systems Research Center, 1992.

[SB94] Manojit Sarkar and Marc H. Brown, *Graphical fisheye views*, Communications of the ACM **37** (1994), no. 12, 73–84.

[SM95] Margaret-Anne D. Storey and Hausi A. Muller, *Graph layout adjustment strategies*, DIMACS Workshop on Graph Drawing, 1995.

[SSTR93] Manojit Sarkar, Scott S. Snibbe, Oren Tversky, and Steven P. Reiss, *Stretching the rubber sheet: A metaphor for visualizing large layouts on small screens*, Proceedings of the ACM Symposium on User Interface Software and Technology, 1993.

[SZaD+93] Doug Schaffer, Zhengping Zuo, Lyn Bartram andJohn Dill, Shelli Dubs, Saul Greenberg, and Mark Roseman, *Comparing fisheye and full-zoom techniques for navigation of hierarchically clustered networks*, Proceedings of Graphics Interface, 1993, pp. 87–96.

[SZG+96] Doug Schaffer, Zhengping Zuo, Saul Greenberg, Lyn Bartram, John Dill, Shelli Dubs, and Mark Roseman, *Navigating hierachically clustered networks through fisheye and full-zoom methods*, ACM Transactions on Computer-Human Interaction **3** (1996), no. 2, 162–188.

[Wil83] L. Williams, *Pyramidal parametrics*, SIGGRAPH, 1983.

Curriculum Vitae

Alan Keahey was born in Huntsville Texas on July 15, 1965, and lived briefly in New Mexico and Tennessee before moving to Canada in 1971. He lived for 12 years in Calgary Alberta, and then moved to Winnipeg Manitoba in 1983, where he alternated between working as a stage lighting technician and getting a B.C.Sc.(Honours) degree from the University of Manitoba. In 1992 he moved to Bloomington Indiana to attend graduate school, where he received his M.S. in Computer Science in 1994, and his Ph.D. (with the completion of this thesis) in 1997. After an extended vacation in Europe, he will be joining Los Alamos National Laboratory as a Technical Staff Member.