Nonlinear Magnification Fields

T. Alan Keahey* and Edward L. Robertson
Department of Computer Science
Indiana University
215 Lindley Hall
Bloomington, IN 47405
{*tkeahey, edrbtsn*}*@cs.indiana.edu*

Abstract

We introduce nonlinear magnification fields as an abstract representation of nonlinear magnification. Methods are provided for converting transformation ("distortion") routines to magnification fields and vice-versa. This new representation provides ease of manipulation and power of expressiveness. We can exploit these on a number of levels: from fine-grained low-level details, to user-interfaces layered on top of our representation, to using the data itself to define the magnification.

1. Introduction

Many approaches have been described in the literature for stretching and distorting spaces to produce effective visualizations of data. Such techniques have been called *polyfocal projection* [KS78], *bifocal display* [SA82], *fisheye views* [Fur86, SB94], *multi-viewpoint perspective display* [MS91], *rubber sheet* [SSTR93], *distortion-oriented presentation* [LA94], *focus + context* [LRP95] and many other terms [PG92, BH94, CCF96]. In [KR96] we introduced the term *nonlinear magnification* to describe the effects common to all of these systems. The basic characteristics of nonlinear magnification are non-occluding in-place magnification which preserves a view of the global context.

In this paper, we will further develop the idea of a theory of nonlinear magnification. In particular, we reduce the concept of nonlinear magnification to a field of scalar magnification values. These nonlinear magnification fields serve as a basis for understanding and comparing the effects of existing techniques, even though their underlying mechanisms may be very different. In addition, we will describe operations which can be performed on nonlinear magnification fields to produce suitable distortions of space for visualization. These operations are entirely view-independent, and do not rely on a physical viewing model or perspective projections.

By expressing magnification as a field of arbitrary scalar values, we provide for much greater expressiveness of magnification and ease of manipulation than is possible using other techniques. In particular, we remove the restriction of having discrete "foci" (centers of magnification), so that fluidly shifting magnifications of arbitrary complexity are now possible. Removing this restriction allows us to factor out magnification complexity from the time required to

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

compute suitable transformation functions, so that computation is *independent* of the complexity of the magnification function.

We will also show how nonlinear magnification fields provide an elegant mechanism for generating visualizations of data which automatically enhance regions of interest. Because nonlinear magnification can now be represented as a simple scalar field, it is possible to use properties of the data being visualized (such as density) to define a magnification that best shows those properties. This ability moves nonlinear magnification beyond simple user-interaction methods, and makes it a first class visualization technique of general applicability.

2. Magnification Fields

When describing nonlinear magnification systems, it is useful to distinguish between *magnification* and *transformation* functions, as first described in [LA94] and expanded upon in [SSTR93, KR96]. The transformation function represents the "distortion" function which stretches the space. The magnification function is the derivative of the transformation function, and represents the magnification values which are implicit in the transformation function. 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 remainder of this section we will describe techniques for accomplishing these conversions, after first introducing some basic notation. [1]

Any magnification employs a transformation function t ($(x', y') = t(x, y)$) which moves points of a rectangular domain $\mathcal{D}$ within a frame. Since we want the magnification to be non-occluding, we require that t is at least C^0 continuous and order-preserving (given $x_1 < x_2$, $(x'_1, y'_2) = t(x_1, y_2)$ and $(x'_2, y'_2) = t(x_2, y_2)$ implies $x'_1 < x'_2$, and similarly for y). For computational purposes, we deal with t only on a $p \times q$ integer grid $\mathcal{G}$, with $g : \mathcal{G} \rightarrow \mathcal{D}$ (g maps the regular grid $\mathcal{G}$ over the domain $\mathcal{D}$), and represent a discrete approximation of t with a quadrilateral grid T ($T = t \circ g$).

A magnification field m is a 2D scalar field of the form $z = m(x, y)$ which gives the regional expansion around a point. As with t, m is represented on $\mathcal{G}$ by a quadrilateral mesh M ($M = m \circ g$). The area-based magnification we use actually corresponds to the square of the linear "power" sometimes used in describing lenses; we do this for computational efficiency. A transformation t corresponds to a magnification m, where $m(x, y) = (\partial t(x, y)/\partial x) \times (\partial t(x, y)/\partial y)$. This is approximated using an area-based function m_c which computes the local magnification for each node in T. In describing computations with magnifications and transformations, notation such as m_α or T_α indicate variations of these functions or their approximations.

2.1. Transformation Grid $\rightarrow$ Magnification Field

Conversion from a given transformation grid T to a magnification mesh M involves numerically computing an approximate derivative of T. The computation begins with an area function a 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 convex hull of the 4-connected neighbours $\{T(i+1, j), T(i-1, j), T(i, j+1), T(i, j-1)\}$. We define C_a to be the constant area associated with any $T(i, j)$ in the untransformed uniform sampling grid. The approximate magnification value for a point $T(i, j)$ is then given by $M(i, j) = m_c(i, j) = a(i, j)/C_a$. More accurate area calculations are possible, such as explicitly finding the area of the four surrounding cells. In practice however, we find that this increase in accuracy does not significantly change the results. Coarser approximations are adequate, as long as the area metric is used consistently throughout the system. Figure 1 shows an example of a transformation 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 with elevation-based shading, somewhat similar to the 3D Pliable Surfaces (3DPS) described in [CCF95].

[1] Although this paper presents results in 2 dimensions, we note that the view-independent nature of the techniques presented here allows for trivial extension to 3 or more dimensions.

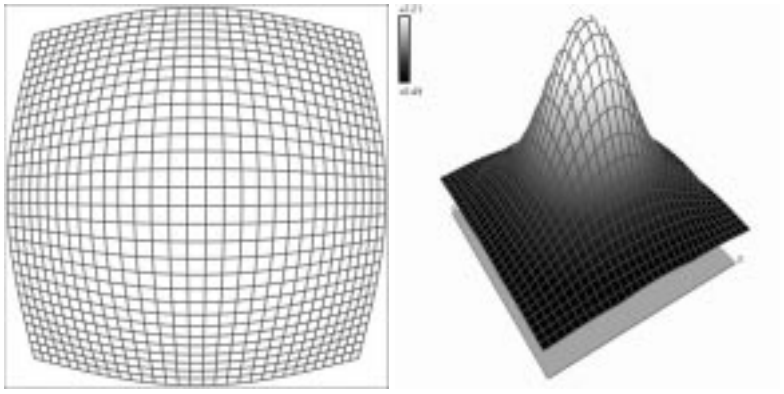

Figure 1. Transformation and Magnification Mesh

One major difference between 3DPS and a magnification landscape defined in this way is that 3DPS does not use true elevation, but rather elevation skewed in the direction of the viewing point. In addition, 3DPS relies on a perspective projection which results in nonuniform changes in viewed scale as uniform steps are taken in elevation. As distance from the viewpoint increases, a constant change in elevation Δz will cause smaller changes in magnification level, as we will illustrate later in Figure 3 (how much smaller is dependent on the current perspective viewing parameters). In contrast, within our magnification system scale and elevation correspond directly.

Figure 2 shows another example illustrating multiple bounded regions and flat magnification, the transformation grid was generated using the techniques described in [KR96]. 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 the 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, CCF95]). 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, notice the curved magnification surfaces caused by the perspective transform's non-uniform correspondence between scale and distance).

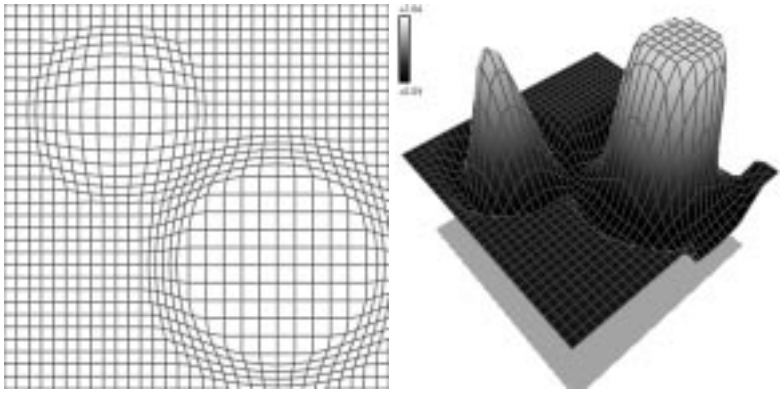

Figure 2. Bounded/Flat Transformation to Magnification Conversion

2.2. Magnification Field $\rightarrow$ Transformation Grid

Given a transformation grid T, it is a relatively straightforward task to find the implicit magnification mesh M associated with it by computing the approximate derivative of T. It is a much more complex task however to construct a

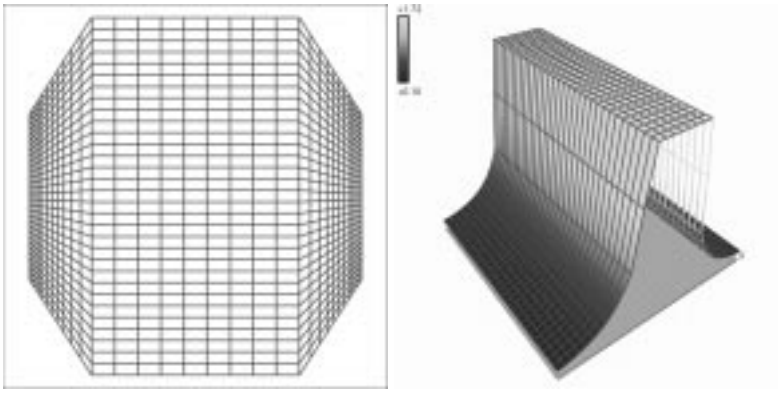

Figure 3. Perspective Wall Transformation and Magnification

suitable transformation grid given a magnification mesh. As pointed out in [CCF95], simple perspective projections of these meshes are not effective because they introduce problems of occlusion.

In general terms, we are required to integrate the magnification mesh values in order to construct an order-preserving transformation grid. 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. We have investigated and developed direct analytic solutions to this problem, but have found these methods to be unsuited to the specific task of generating nonlinear magnification transformations. Some of the major problems which we have observed with the direct approaches are: 1) bounded regions of magnification in M should produce bounded regions of transformation in T to preserve a static context, 2) the transformation should be symmetric and centered around magnification maxima, and not constructed relative to some arbitrary boundary of the domain, and 3) 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 flat magnification.

In order to address these problems we have developed an iterative method which provides an approximate numerical solution to the integration problem. By dealing with a localized basis for computation, we are able to simply and directly control the overall behaviour of the algorithm to produce the desired final result. Given a specified magnification mesh M_S, we want to compute a transformation grid T_C which yields a similar implicit magnification mesh M_C. We define $M_E = (M_S - M_C)$ to be the error mesh, which reflects the difference between the specified magnification and the magnification generated by the current T_C. 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}. We similarly join the M_E values to T_C giving M_{Ev}. M_{Cv} and M_{Ev} are used for visualization only, and are not used in any internal calculations.

Conceptually, our algorithm is straightforward. We begin with a uniform T_C representing the identity transformation. 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 $T_C(i, j)$ away a little bit from $T_C(i, j)$. Conversely if $M_E(i, j) < 0$ then we pull the neighbours of $T_C(i, j)$ 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. Figure 4 shows an example of the operation of this algorithm over a few time steps (see also Color Plate A); and Figure 5 shows how our method handles multiple, bounded and flat regions of magnification. There are a number of parameters and issues to explore in this algorithm, which we discuss below; first however we should point out that not all possible magnification specifications will have a solution. The most obvious example is a mesh specifying $2\times$ magnification across the entire field. This cannot possibly be satisfied while maintaining the desired properties of non-occluding in-place magnification. We do not expect our algorithm to converge on a solution in all such degenerate

cases, although we are pleased when it manages to compute a reasonable compromise anyway.

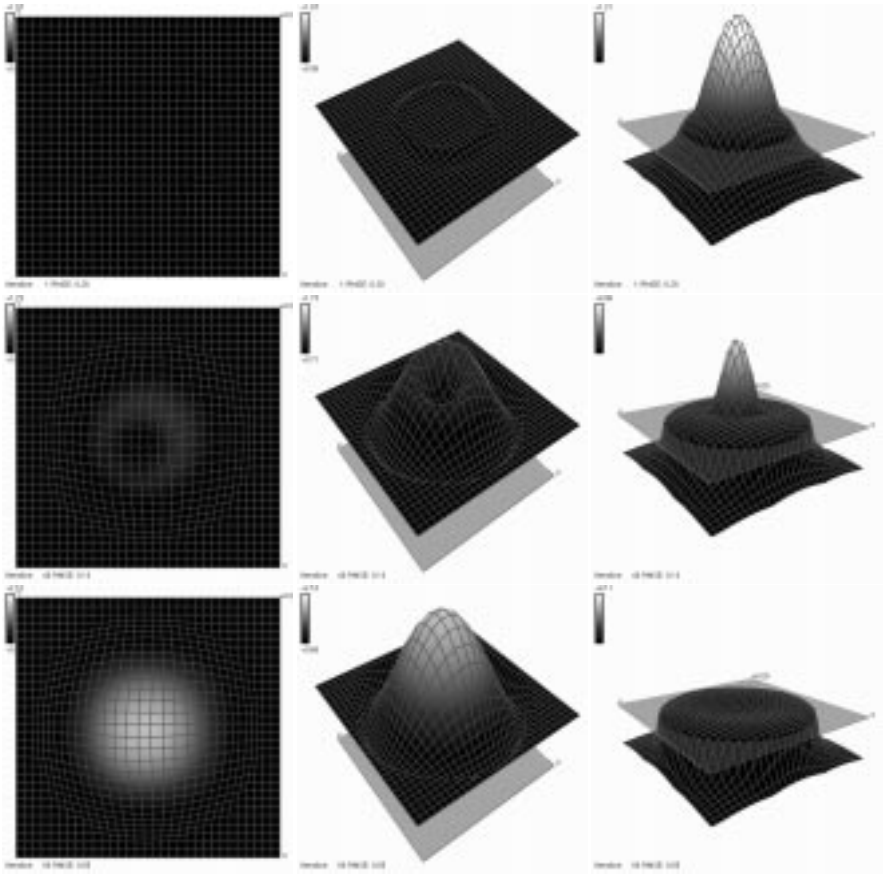

Figure 4. T_C**,** M_{Cv} **and** M_{Ev} **on Iteration 1,40,80 using** M_S **from Figure 1**

As mentioned previously, many different area metrics can be used in these methods. The area metric used will determine which neighbours should be displaced in our algorithm (*i.e.* if the area metric is 4-connected, then the algorithm should only displace the 4-connected neighbours). 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 thrash and possibly not converge at all. We typically use $C_r = 0.3$.

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. Similarly for $M_E(i,j) < 0$, we weight the displacements to pull more distant neighbours a greater distance than closer neighbours.

Our algorithm converges independent of the complexity of the magnification field, where convergence is measured by root mean squared error: *RMSE*= $\sqrt{\sum_{i=1}^{p}\sum_{j=1}^{q}(M_E(i,j))^2}$. The primary determining factor of convergence speed is the volume of specified magnification, or more precisely the volume of error in M_E. There are a number of param-

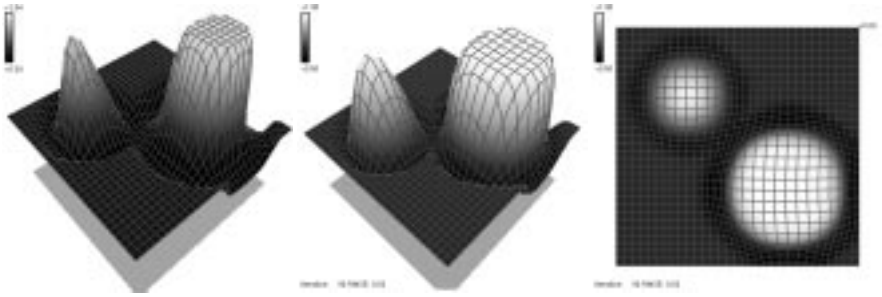

Figure 5. M_S, M_{Cv} and T_C Illustrating Multiple and Flat Regions

eters which we can use to tune the performance of our algorithm based on speed/accuracy tradeoffs. First we create an error clipping constant C_e with a default value of 0, then our algorithm ignores nodes where $M_E(i,j) < C_e$. This causes it to converge faster because it does not have to compress areas whose implicit magnification is greater than its specified magnification. The result of this is that some transformation grids which are used to construct magnification field specifications may not be exactly reconstructed using this iterative magnification to transformation conversion (particularly on unbounded transformations). The only difference however 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, eliminating dead screen regions which were outside the range of the original transformation grid (compare the final step in Figure 4 with Figure 1). More significant is the fact that by reducing the emphasis on demagnification in this way, we eliminate the deadlock condition than can be caused by some degenerate cases. An example of where this can occur is when a region of very low magnification is surrounded by regions of high magnification (a doughnut shape) in such a way that the conditions of the regions conflict with each other so that our algorithm can not satisfy both. We make the assumption that regions of high magnification (and high error) should take higher priority in the algorithm, and this solves the problem of conflict (it is worth noting that the methods which we present here can also be used to emphasize areas demagnification simply by changing the error metric to emphasize negative error values rather than positive ones). When this error clipping method is used, we redefine our error measure as: *RMSE*= $\sqrt{\sum_{i=1}^{p}\sum_{j=1}^{q}(Max(0, M_E(i,j) - C_e))^2}$, and say 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.

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. By carefully adjusting C_m and C_e for the particular application, dramatic 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. Mesh resolution is also a significant factor in the performance of our algorithm. High resolution meshes are able to compute finer detail that lower resolution meshes, but generally require greater computation time. Thus mesh resolution is another parameter in our systems which can be used to tune results. Table 1 summarizes the effect of adjusting these parameters [2].

2.3. Magnification Field $\leftrightarrow$ Transformation Grid

Given a transformation grid T and its associated magnification mesh M, we can produce a blend of T and M which allows the viewer to visualize the transformation as smoothly shifting into 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

[2] Timings were obtained on a 174 MHz MIPS R10000 CPU.

$s \times t$	C_m	C_e	Iterations	Time (s)
32×32	0.0	0.00	59	0.45
		0.25	39	0.12
	0.75	0.00	59	0.24
		0.25	39	0.08
24×24	0.0	0.00	33	0.09
		0.25	19	0.04
	0.75	0.00	33	0.07
		0.25	19	0.02

Table 1. Performance Using M_S from Figure 1

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 $t = 2\theta/\pi$, then the interpolation is given as $M' = (1-t)T + tM$. The effect of this is an aerial view of the transformation grid which can be "pulled down" to view as a landscape of magnification values. Figure 6 shows a few snapshots of this operation, refer to the accompanying video for a clearer view of this effect.

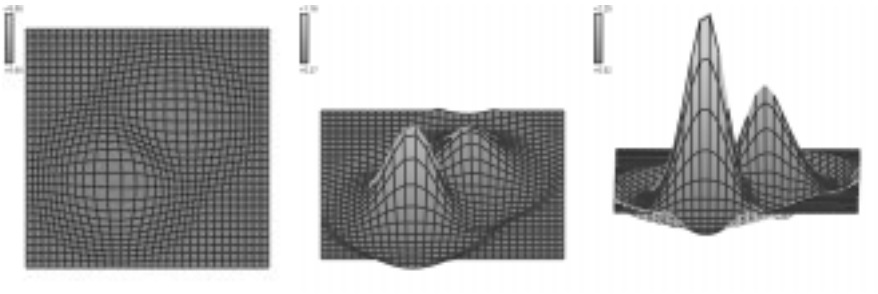

Figure 6. Blending T Into A Magnification Landscape M

3. 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 function. We refer to systems which rely on the side-effects of transformation functions to produce nonlinear magnification as *transformation-based* systems; in contrast the system we present here is a *magnification-based* system. Direct manipulation of the magnification mesh makes for a system which is both simpler and more powerful 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 2.2 automate the task of constructing 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 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.

3.1. Node-Level Specification

At the lowest level, we can control the magnification mesh on a node by node basis. For demonstration purposes, we have created 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 us with 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 arbitrary regions of the domain to be excluded from the transformations (*i.e.* unmagnified). In addition, it now becomes a trivial matter to constrain the transformation to any arbitrary bounded domain (including concave domains) simply by locking those nodes which define that bounded domain (see Figure 7). The examples in this paper used locked nodes on the mesh perimeter to ensure that the transformed grid would still fit in the original rectangular sampling area.

A major feature of this locking mechanism is that in many cases specifying arbitrary bounded regions of magnification (or non-magnification) actually *reduces* the computation required (assuming degenerate cases are not introduced). Other nonlinear magnification systems featuring bounded regions of magnification [SSTR93, CCF95, KR96] require additional computation and program complexity to ensure boundedness of magnification either within a region of the domain or within the viewing frustum. In contrast, the locking mechanism which we present here allows us to obtain arbitrary bounded regions for "less than nothing" in computational cost in most cases. The additional program complexity is a trivial boolean flag check for each node in the iterative conversion process.

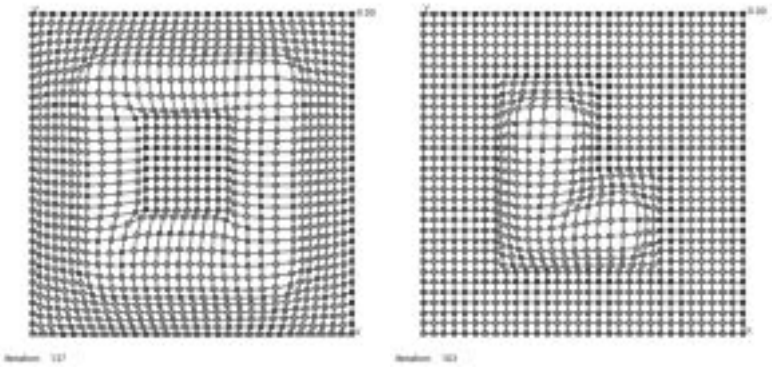

Figure 7. Locking and Bounding at the Node Level

3.2. 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 existing transformation-based systems. Given a transformation grid T having an implicit magnification mesh M, it is a simple matter to compute the inverse mesh M^{-1}, and then find the approximate inverse transformation T^{-1} (see Figure 8). 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: $M(i,j) = tM_a(i,j) + (1-t)M_b(i,j)\ \ (0 \leq t \leq 1)$, 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-lock 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.

3.3. User-Level Interface

The expressiveness and implementation-independent nature of our 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 custom magnification tools and techniques which are tailor made to specific tasks.

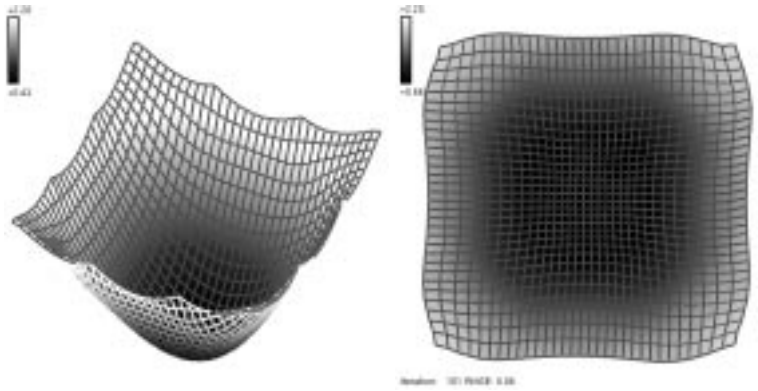

Figure 8. Inverse of Figure 1 M and T

We have just begun to explore the possibilities of layering interfaces on top of our general magnification field techniques. Perhaps the simplest interface involves construction of a discrete rectangular "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 9 and Color Plates B and C). 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 9). 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.

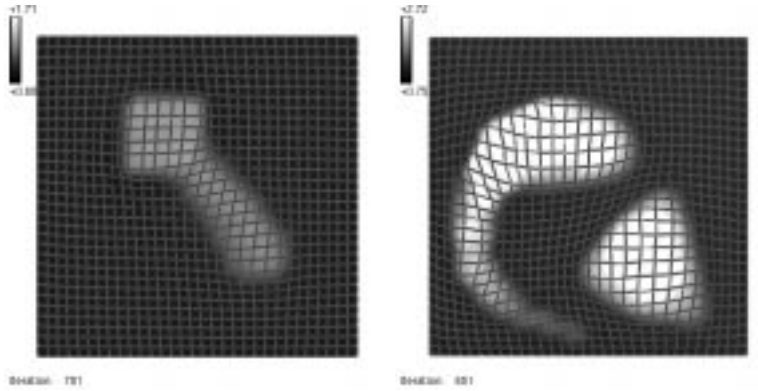

Figure 9. Magnifying Trail and Brush

3.4. Application-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 paper. Certainly one significant bridge – perhaps the most significant one – between these levels is to use the data to control magnification.

One major reason for implementing transformation 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 10 shows a simulated air traffic control system where the scale is automatically magnified in regions of higher traffic density[3].

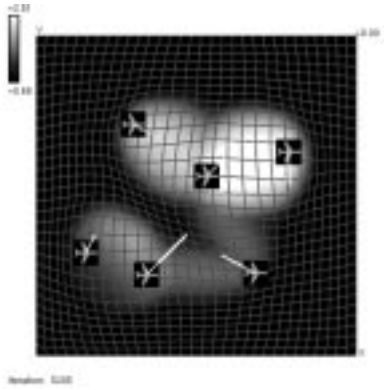

Figure 10. Using Density to Define ATC Magnification

The efforts we have engaged in thus far are a mere beginning of dealing with data directed magnifications. In particular, they are aimed at providing tools necessary for in-depth study of the human interaction factors involved but not at conducting those human factors experiments. A simple example where substantially more work will be required is the shape of the magnified regions. While the computational process smooths the boundaries of a "gerrymandered" magnification grid, the resulting region may still clash with the viewer's intuition or esthetics.

4. 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 serve as the basis for a constructive theory. One of the goals of this paper is 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

[3]The authors (who fly fairly frequently) would suggest greater human-factors studies be carried out before this technique is used in an actual control tower.

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.

3D pliable surfaces (3DPS) [CCF95] resemble the magnification meshes presented here, particularly in the use of elevation information to produce shading cues indicating regions of magnification. There are more fundamental differences than similarities however. We have already described the difference in the way which the two systems correspond elevation to magnification in section 2.1. 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 the view frustum is not inherent, and requires additional constraints on parameters, whereas by its very nature our iterative system guarantees non-occlusion and confinement to any size or shape of domain. More fundamentally, 3DPS is a view-dependent system which is tied very closely to its own specific implementation of a physical model using perspective projection. We have shown in comparison how our system is less implementation dependent, requiring no physically based model, and that the concepts and techniques we have developed are applicable to a broad range of existing nonlinear magnification systems.

5. Conclusions

Nonlinear magnification fields provide an abstract representation for dealing with nonlinear magnification systems. We have shown how the magnification effects of other continuous nonlinear magnification systems can be examined and compared by constructing implicit magnification fields from their transformation routines. In the other direction, the iterative method which we described allows construction of a transformation from an arbitrary magnification field specification. Our method is simple and effective, even on complex magnification fields having multiple maxima and regions of flat magnification, and can be easily tuned with a number of parameters to control overall performance.

Our abstract field representation is expressive and easy to manipulate. By removing the restrictions of view dependence and explicit foci, our system provides a natural and intuitive means of specifying magnification which does not rely on the side effects of complexly interacting transformation functions. 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 note is the ability to use properties of the data itself to define the magnification fields best suited to visualizing that data.

6. Further Work

Our iterative method for converting magnification meshes to transformation grids provides good results, but can be improved on. We plan to investigate the use of multi-resolution grids such as quad-trees to provide finer control over variable resolution magnification. The representation may also prove to be useful for progressive refinement techniques to increase speed and interactivity. More work can also be done on providing effective interfaces to our low-level routines in a way which allows us to take advantage of the power and flexibility offered by this system. This work is proceeding on two levels: constructing user interfaces for interactive application and exploring further how properties of data can best be used to define magnifications.

References

[BH94] 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.

[CCF95] 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.

[CCF96] 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.

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

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

[KR96] 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.

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

[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.

[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.

[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.

[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.

[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.

[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.

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

[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.