Nonlinear Magnification Fields [*] [†]

T. Alan Keahey[‡] Edward L. Robertson
Department of Computer Science
Indiana University, Bloomington, IN 47405
{*tkeahey,edrbtsn*}*@cs.indiana.edu*

Abstract

We introduce nonlinear magnification fields as an abstract representation of nonlinear magnification, providing methods for converting transformation routines to magnification fields and vice-versa. This new representation provides ease of manipulation and power of expression. By removing the restrictions of explicit foci and allowing precise specification of magnification values, we can achieve magnification effects which were not previously possible. Of particular interest are techniques we introduce for expressing complex and subtle magnification effects through magnification brushing, and allowing intrinsic properties of the data being visualized to create data-driven magnifications.

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*[4], *bifocal display*[14], *fisheye views*[2, 12], *multi-viewpoint perspective display*[10], *rubber sheet*[13], *distortion-oriented presentation*[8] and *focus + context*[7]. In [5] 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. During our research on the work described in [5] there were a number of issues which we encountered. A brief discussion of some of these issues will provide context and motivation for the results presented in the remainder of this paper.

The focus of our work in [5] was to define a nonlinear magnification system using simple modular components joined together to create complex magnification effects. In doing this, we distinguished between linear and nonlinear magnification transformations and illustrated how the best properties of each could be exploited by providing simple methods to combine these within a single transformation, and by introducing constrained transformation domains which remain invariant even when the magnification or compression changes. We extended the work on 1D piecewise functions in [8, 13], creating 1D piecewise linear transformations as efficient and expressive alternatives to conventional continuous 1D transformation functions.Then we introduced 2D piecewise linear transformations which provide approximations to complex sequences of transformations. We described how 1D piecewise functions could be easily manipulated through changes to either the magnification or transformation function. We then raised the question of how such manipulations could be performed on 2D piecewise linear transformations. This paper will provide answers to that question.

Although our 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 properties which our modular system still shares with existing nonlinear magnification systems. These issues have lead us to look for a more general schema for defining and implementing 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 the addition of more and/or complex lenses, thus increasing the computational complexity of computing the overall transformation.

A second difficulty encountered with existing systems for nonlinear magnification 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.

[*] http://www.cs.indiana.edu/hyplan/tkeahey/research/nlm/nlm.html

[†] Copyright 1997 IEEE. Published in the Proceedings of IEEE Visualisation '97, Information Visualization Symposium, October 20-21 1997, Phoenix, Arizona. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works, must be obtained from the IEEE.

[‡] This work supported by US Dept of Education Award P200A502367.

In this paper 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 realizing the effects of existing techniques, even though their underlying mechanisms may be very different. Second, they directly define space transforming visualizations and can be operated on in computationally efficient and conceptually effective ways, thus yielding powerful visualization tools.

Expressing 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 describe 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.

In brief, the work described in this paper makes the following major contributions:

- introduces a general method for converting complex transformations of two or more dimensions to scalar magnification fields
- introduces an iterative method for converting scalar magnification fields to transformations
- introduces *magnification brushing* as an user-interface technique allowing creation of complex and subtle magnification effects
- introduces *data-driven magnification* as a technique which allows properties of the data to directly define magnifications for viewing that same data

2. Magnification Fields

When describing nonlinear magnification systems, it is useful to distinguish between *magnification* and *transformation* functions, as first described in [8]. 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. 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 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 $(x'_1, y'_1) = t(x_1, y_1)$ and $(x'_2, y'_2) = t(x_2, y_2)$, $x_1 < x_2$ 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 ($T = t \circ g$, where '$\circ$' represents composition).

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 ($M = m \circ g$). 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. For computational efficiency, the area-based magnification we use actually corresponds to the square of the linear "power" sometimes used in describing lenses. In describing computations with magnifications and transformations, notation such as m_c or T_C indicate variations of these functions or their approximations.

2.1. Transformation Grid → 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. The surfaces described in 3DPS [1] may appear to be similar to this land-

[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.

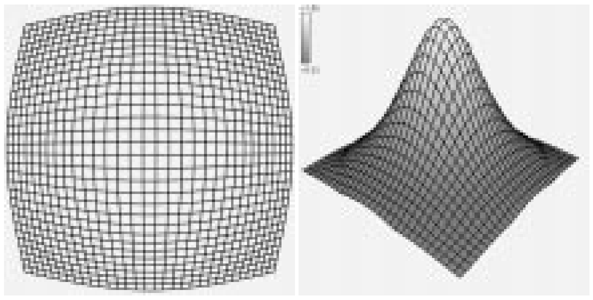

Figure 1. Transformation and Magnification

scape representation, 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 view-dependent 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 we present here shades regions of magnification simply by mapping elevation values into a color/intensity ramp.

Figure 2 shows another example illustrating multiple bounded regions and linear magnification; the transformation grid was generated using the techniques described in [5]. 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 [9], which is representative of the class of nonlinear magnification systems that are based on a perspective projection of 3D surfaces. 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). These examples show how transformation and magnification functions can now be tightly coupled across entire domains even for complex transformations.

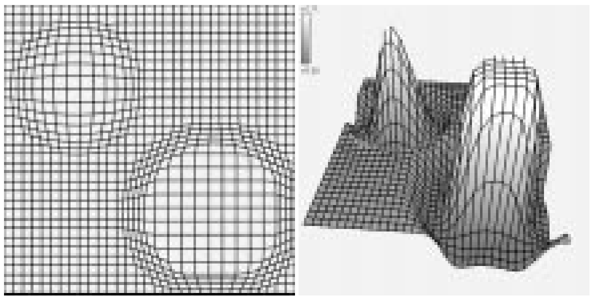

Figure 2. Complex Magnification Field

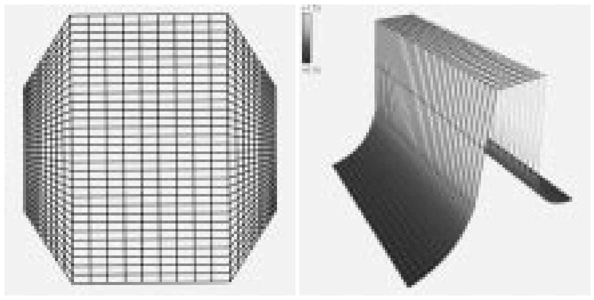

Figure 3. Perspective Wall Magnification

2.2. Magnification Field → Transformation Grid

While it is a relatively straightforward task to find the implicit magnification mesh M associated with a transformation grid T by computing the approximate derivative; it is a much more complex task to construct a suitable transformation grid given a magnification mesh. In general terms, we want 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 methods to solve 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 linear magnification.

In order to address these problems we have developed an iterative method which provides a 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. 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 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. 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 initialize T_C to 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 iterations (using M_S from Figure 1); and Figure 5 shows how our method handles the multiple, bounded and linear regions of magnification specified in Figure 2. There are a number of parameters and issues to explore in this algorithm, which we discuss below.

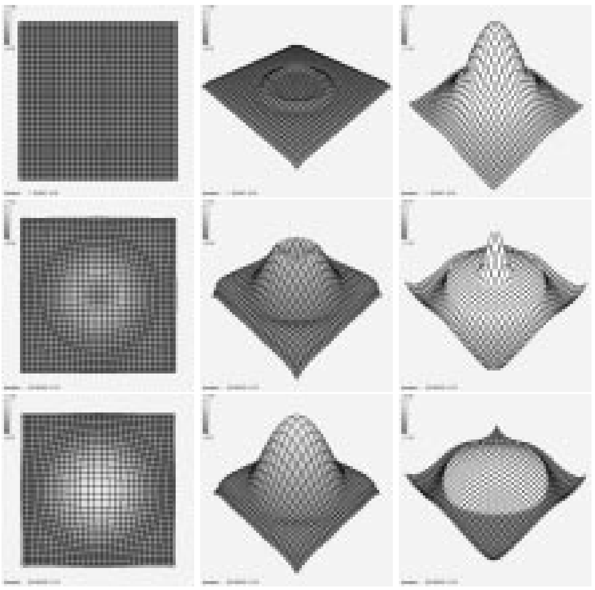

Figure 4. T_C**,** M_{Cv} **and** M_{Ev} **on Iteration 1,40,80**

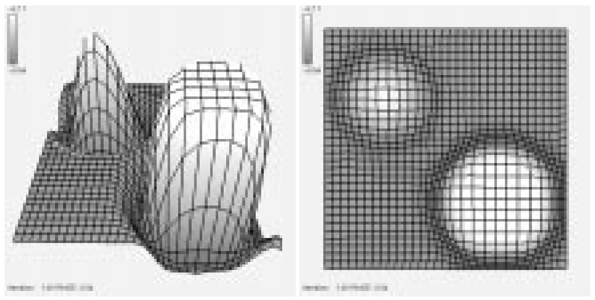

Figure 5. M_{Cv} **and** T_C **Computed from Figure 2**

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 parameters 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 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. Color Plate A shows a few iterations on an input identical to that used for Figure 4, except that error clipping with $C_e = 0$ is used on the color plate. 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 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.

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 robust than magnification clipping 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. Mesh resolution is also a significant factor in the performance of our algorithm. High resolution meshes are able to compute finer detail than 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].

$p \times q$	C_m	C_e	Iterations	Time (s)
32×32	-	-	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
24×24	-	-	98	0.298
		0.00	40	0.072
		0.25	28	0.030
	0.75	-	40	0.068
		0.00	40	0.058
		0.25	28	0.022

Table 1. Performance Using M_S from Figure 1

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 reasonable compromise for them. We describe the degenerate cases below, along with how they can be effectively dealt with. First however we point out 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 must cause a corresponding compression at some other region. Another related type of degenerate case involves conflicting regions of magnification. For example when a region of very low magnification is surrounded by regions of high 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. 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.

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. 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 [9, 11, 1]. 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 we present 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 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 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

[2]Timings were obtained on a 195 MHz MIPS R10000 CPU.

direct magnification specification and end with a transformation reflecting that specification. We intend to illustrate through these examples that "magnification" is a more intuitive and useful interface concept than "distortion". Distortion is not the goal of our system but only a by-product, although our 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 gradient of the magnification field, as shown in Figure 6 and Color Plate B. This clearly reveals the derivative nature of distortion.

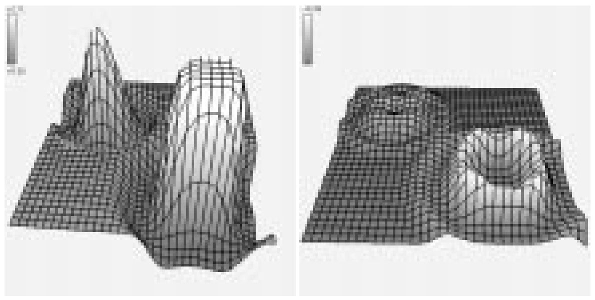

Figure 6. Magnification/Distortion from Figure 2

3.1. Node-Level

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 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 7; 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 7). Some of the examples in this paper 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 [6, 5][3], 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.

[3]The "distortion control" presented in [1] is not equivalent to these, as it does not provide invariant boundaries that are independent of the magnification/distortion parameters.

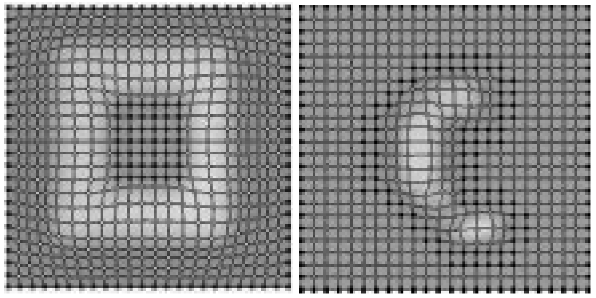

Figure 7. Excluded and Bounded Regions

3.2. Mesh-Level

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 inverse mesh M^{-1}, and then find the 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: $M(i,j) = dM_a(i,j) + (1-d)M_b(i,j) \quad (0 \le d \le 1)$, 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.

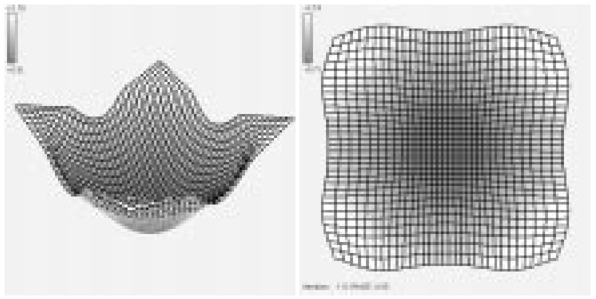

Figure 8. Inverse of Figure 1 M^{-1} and T^{-1}

3.3. User-Level

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 magnification tools and techniques which are customized to specific tasks.

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 "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 Plate 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.

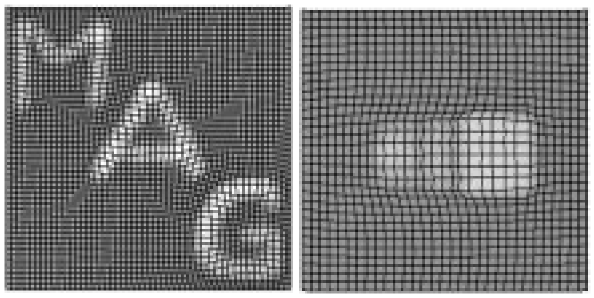
Figure 9. Magnifying Brush and Trail

3.4. Data-Level

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. 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 10 and Color Plate D show a simulated air traffic control system where regions of higher traffic density are automatically magnified [4].

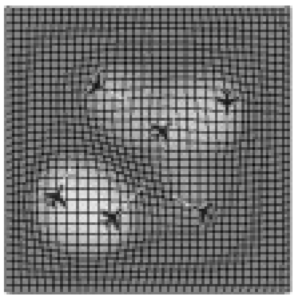
Figure 10. Data-Driven Magnification

4. Related Work

Leung and Apperley [8] 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. One of the goals of the current 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 [3] 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

[4]The authors (who fly frequently) would suggest further human-factors studies be carried out before this technique is tried in actual control towers.

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.

We have already described significant differences between 3DPS [1] and our system 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 fixed regions is not inherent, and requires additional (unspecified) constraints on parameters, whereas by its very nature our iterative system 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 our system is less implementation dependent, and the concepts and techniques can be directly applied to a broad range of visualization and nonlinear magnification systems.

5. Conclusions and Further Work

Nonlinear magnification fields provide a natural 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 transformations, providing a consistent mapping between complex transformation and magnification functions. Going in the other direction, the iterative method which we present allows construction of a transformation from an arbitrary magnification field specification. Our method is simple and effective, even on complex fields having multiple maxima, bounded regions, and areas of linear magnification. A number of parameters can be easily tuned to control overall performance.

Our abstract magnification 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 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.

We are investigating the use of multi-resolution methods to increase the speed and interactivity of our iterative method. More work is also being done on providing effective interfaces to our low-level routines in a way which takes advantage of the power and flexibility of 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 create data-driven magnifications.

References

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

[2] G. W. Furnas. Generalized fisheye views. *Human Factors in Computing Systems, CHI ’86*, pages 16–23, Apr. 1986.

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

[4] N. Kadmon and E. Shlomi. A polyfocal projection for statistical surfaces. *The Cartographic Journal*, 15(1):36–41, June 1978.

[5] T. A. Keahey and E. L. Robertson. Techniques for non-linear magnification transformations. In *Proceedings of the IEEE Symposium on Information Visualization, IEEE Visualization*, pages 38–45, Oct. 1996.

[6] T. A. Keahey and E. L. Robertson. Techniques for nonlinear magnification transformations. Technical Report 455, Department of Computer Science, Indiana University, Mar. 1996. Draft submission of InfoVis’96 paper.

[7] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the ACM Conference on Computer Human Interaction*, 1995.

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

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

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

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

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

[13] M. Sarkar, S. S. Snibbe, O. Tversky, and S. P. Reiss. Stretching the rubber sheet: A metaphor for visualizing large layouts on small screens. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 1993.

[14] R. Spence and M. Apperley. Data-base navigation: An office environment for the professional. *Behaviour and Information Technology*, 1(1):43–54, 1982.

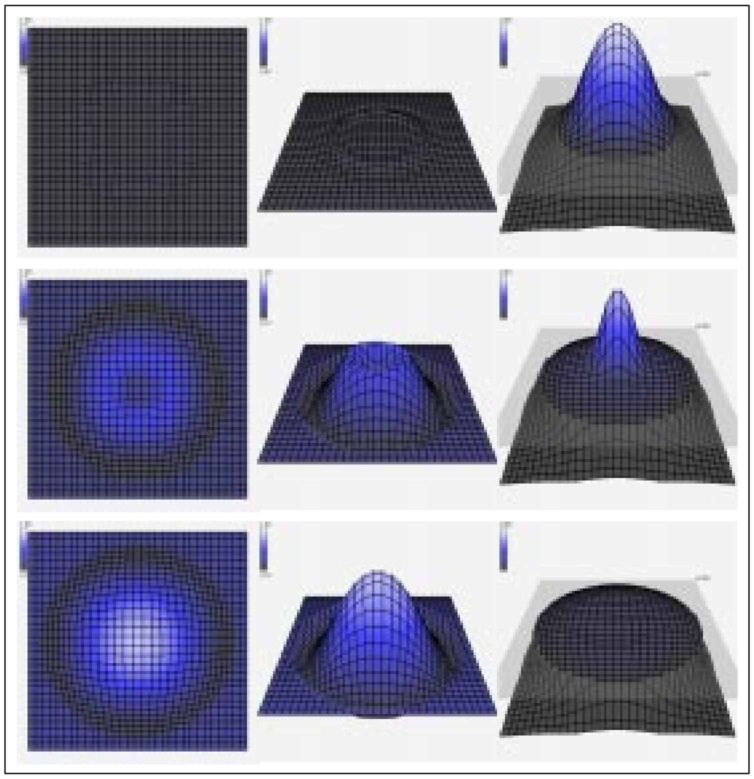

Plate A:
Convergence with error clipping
Iteration 1, 40, 80 (T_C , M_{Cv} , M_{Ev})

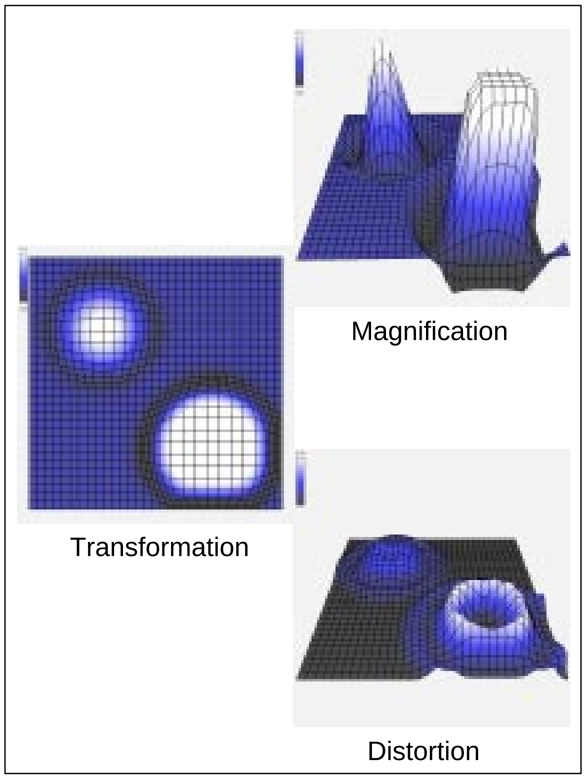

Plate B:
Relation of transformation, magnification and distortion

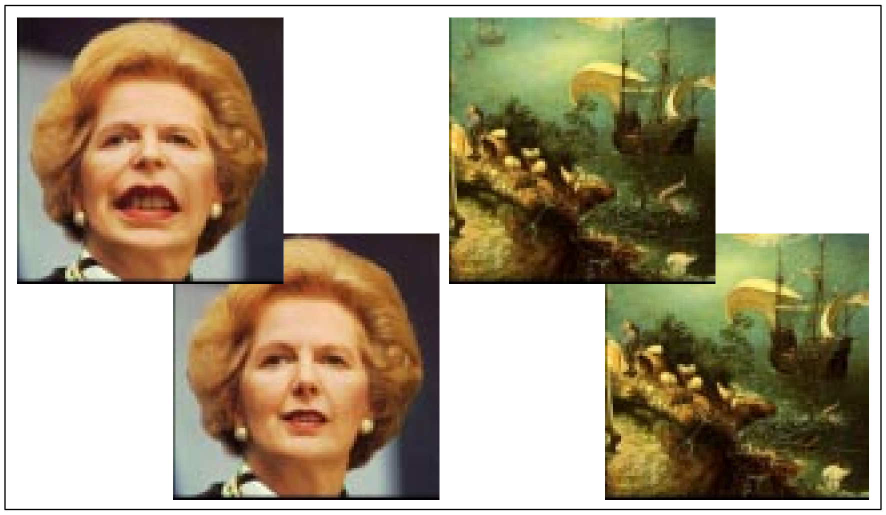

Plate C:
Magnification Brushing

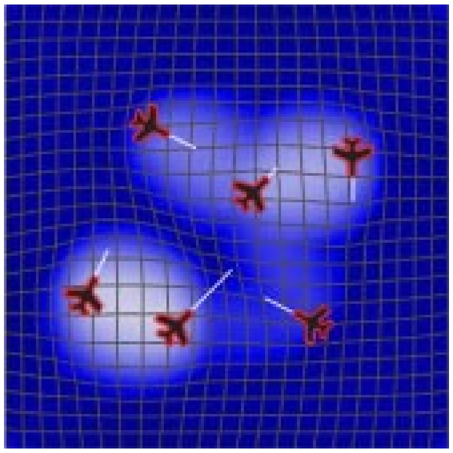

Plate D:
Data−Driven Magnification