SPACE-SCALE DIAGRAMS: UNDERSTANDING MULTISCALE INTERFACES

George W. Furnas
Bellcore, 445 South Street
Morristown, NJ 07962-1910
(201) 829-4289
gwf@bellcore.com

*Benjamin B. Bederson**
Bellcore, 445 South Street
Morristown, NJ 07962-1910
(201) 829-4871
bederson@bellcore.com

ABSTRACT

Big information worlds cause big problems for interfaces. There is too much to see. They are hard to navigate. An armada of techniques has been proposed to present the many scales of information needed. Space-scale diagrams provide a framework for much of this work. By representing both a spatial world and its different magnifications explicitly, the diagrams allow the direct visualization and analysis of important scale related issues for interfaces.

KEYWORDS: Zoom views, multiscale interfaces, fisheye views, information visualization, GIS; visualization, user interface components; formal methods, design rationale.

INTRODUCTION

For more than a decade there have been efforts to devise satisfactory techniques for viewing very large information worlds. (See, for example, [6] and [9] for recent reviews and analyses). The list of techniques for viewing 2D layouts alone is quite long: the Spatial Data Management System [3], Bifocal Display[1], Fisheye Views [4][12], Perspective Wall [8], the Document Lens [11], Pad [10], and Pad++ [2], the MacroScope[7], and many others.

Central to most of these 2D techniques is a notion of what might be called multiscale viewing. Information objects and the structure embedding them can be displayed at many different magnifications, or scale. An interface technique is devised that allows users to manipulate which objects, or which part of the structure will be shown at what scale. The scale may be constant and manipulated over time as with a zoom metaphor, or varying over a single view as in the distortion techniques (e.g., fisheye or bifocal metaphor). In either case, the basic assumption is that by moving through space and changing scale the users can get an integrated notion of a very large structure and its contents, and navigate through it in ways effective for their tasks.

This paper introduces *space-scale* diagrams as a technique for understanding such multiscale interfaces. These diagrams make scale an explicit dimension of the representation, so that its place in the interface and interactions can be visualized, and better analyzed. We are finding the diagrams useful for trying to understand such interfaces geometrically, for guiding the design of code, and perhaps even as interfaces to authoring systems for multiscale information.

This paper will first present the necessary material for understanding the basic diagram and its properties. Subsequent sections will then use that material to show several examples of their uses.

THE SPACE-SCALE DIAGRAM

The basic diagram concepts

The basic idea of a space-scale diagram is quite simple. Consider, for example, a square 2D picture (Figure 1a). The space-scale diagram for this picture would be obtained by creating many copies of the original 2-D picture, one at each possible magnification, and stacking them up to form an inverted pyramid (Figure 1b). While the horizontal axes rep-

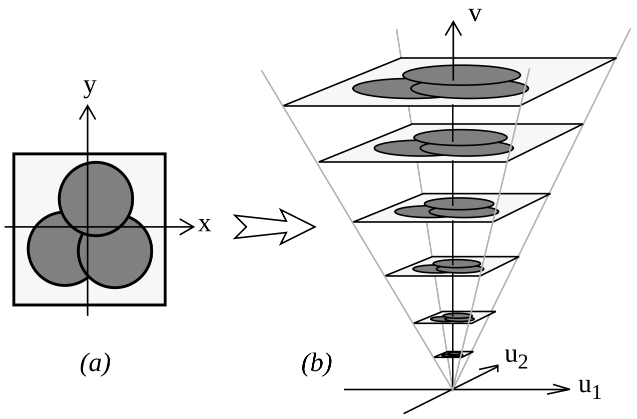

Figure 1. *The basic construction of a Space-Scale diagram from a 2D picture.*

*Current address: bederson@cs.unm.edu, Computer Science Department, University of New Mexico, Albuquerque, NM 87131

This work was paid for in part by ARPA grant N66001-94-C-6039.

resent the original spatial dimensions, the vertical axis represents scale, i.e. the magnification of the picture at that level. In theory, this representation is continuous and infinite: all magnifications appear from 0 to infinity, and the "picture" may be a whole 2D plane if needed.

Before we can discuss the various uses of these diagrams, three basic properties must be described. Note first that a user's viewing window can be represented as a fixed-size horizontal rectangle which, when moved through the 3D space-scale diagram, yields exactly all the possible pan and zoom views of the original 2D surface (Figure 2). This property is useful for studying pan and zoom interactions in highly zoomable interfaces like Pad and Pad++ [2][10].

Secondly, note that a point in the original picture becomes a ray in this space-scale diagram. The ray starts at the origin and goes through the corresponding point in the continuous set of all possible magnifications of the picture (Figure 3). We call these the *great rays* of the diagram. As a result, regions of the 2D picture becomes generalized cones in the diagram. For example, circles become circular cones and squares become square cones.

A third property follows from the fact that typically the properties of the original 2D picture (e.g., its geometry) are considered invariant under moving the origin of the 2D coordinate system. In the space-scale diagrams, such a change of origin corresponds to a "shear" (Figure 4), i.e., sliding all the horizontal layers linearly so as to make a different line become vertical. Thus the only meaningful contents of the space-scale diagram are properties invariant under such a shear. So, in using the diagrams to understand and analyze multiscale interface paradigms, one must not read too much into them, always remembering this third property, that if the origin is not special in the purely spatial representation, only conclusions invariant under shear are valid in the space-scale diagram.

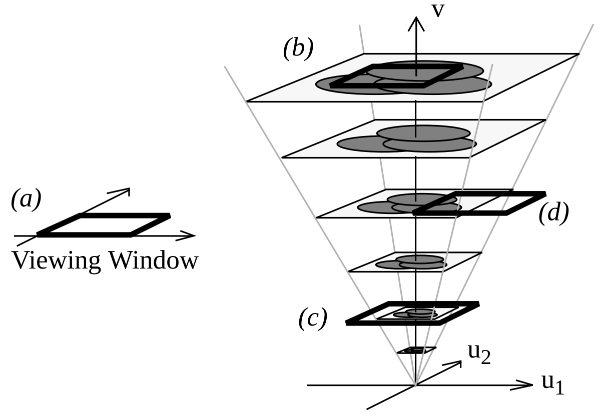

Figure 2. *The viewing window (a) is shifted rigidly around the 3D diagram to obtain all possible pan/zoom views of the original 2D surface, e.g., (b) a zoomed in view of the circle overlap, (c) a zoomed out view including the entire original picture, and (d) a shifted view of a part of the picture.*

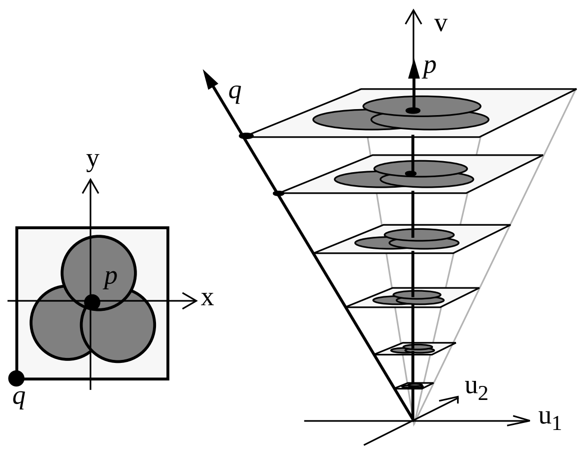

Figure 3. *Points like p and q in the original 2D surface become corresponding "great rays" p and q in the space-scale diagram. (The circles in the picture therefore become cones in the diagram, etc.)*

Now that the basic concepts and properties of space-scale diagrams have been introduced by the detailed Figures 1-4 we can make a simplification. Those figures have been three dimensional, comprising two dimensions of space and one of scale ("2+1D"). Substantial understanding may be gained, however, from the much simpler two-dimensional versions, comprising one dimension of space and one dimension of scale ("1+1D"). In the 1+1D diagram, the spatial world is 1D and so a viewing window is a line segment that can be moved around the diagram to represent different pan and zoom posi-

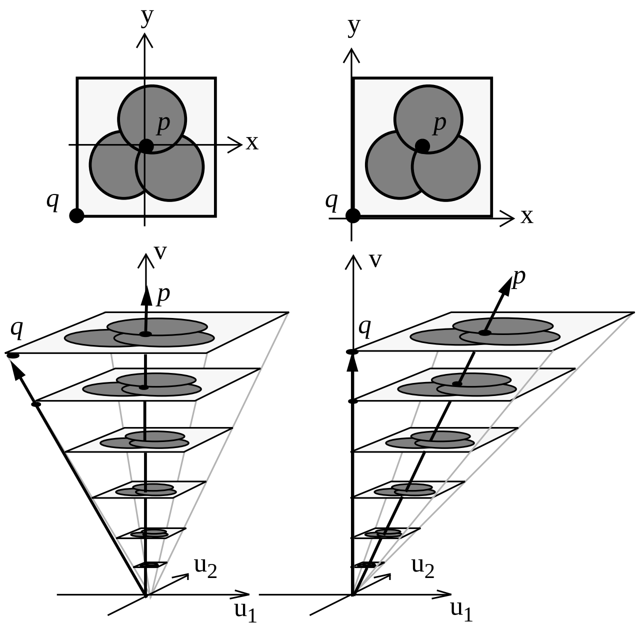

Figure 4. *Shear invariance. Shifting the origin in the 2D picture from p to q corresponds to shearing the layers of the diagram so the q line becomes vertical. Each layer is unchanged, and great rays remain straight. Only those conclusions which remain true under all such shears are valid.*

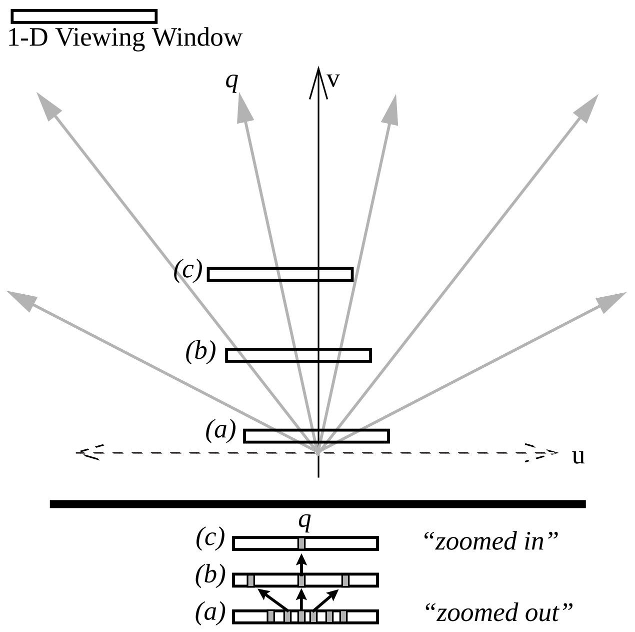

Figure 5. *A "1+1D" space-scale diagram has one spatial dimension, u, and one scale dimension, v. The six great rays here correspond to six points in a 1D spatial world, put together at all magnifications. The viewing window, like the space itself, is one dimensional, and is shown as a narrow slit with the corresponding 1-D window view being visible through the slit. Thus the sequence of views (a), (b), (c) begins with a view of all six points, and then zooms in on the point q. The views, (a), (b), (c) are redrawn at bottom to show the image at those points.*

tions. It is convenient to show the window as a narrow slit, so that looking through it shows the corresponding 1D view. Figure 5 shows one such diagram illustrating a sequence of three zoomed views.

The basic math.

It is helpful to characterize these diagrams mathematically. This will allow us to use analytic geometry along with the diagrams to analyze multiscale interfaces, and also will allow us to map conclusions back into the computer programs that implement them.

The mathematical characterization is simple. Let the pair *(x, z)* denote the point *x* in the original picture considered magnified by the multiplicative scale factor *z*. We define any such *(x, z)* to correspond to the point *(u, v)* in the space-scale diagram where $u=xz$ and $v=z$. (This second trivial equation is needed to make the space-scale coordinates distinct, and because there are other versions of space-scale diagrams, e.g., where $v=\log(z)$). Conversely, of course, a point *(u, v)* in the space-scale diagram corresponds to *(x, z)*, i.e., a point x in the original diagram magnified by a factor *z*, where $x=u/v$, and $z=v$. (The notation is a bit informal, in that *x* and *u* are single coordinates in the 1+1D version of the diagrams, but a sequence of two coordinates in the 2+1D version.)

A few words are in order about the XZ vs. UV characterizations. The *(x,z)* notation can be considered a world-based coordinate system. It is important in interface implementation because typically a world being rendered in a multiscale viewer is stored internally in some fixed canonical coordinate system (denoted with *x*'s). The magnification parameter, *z*, is used in the rendering process. Technically one could define a type of space-scale diagram that plots the set of all *(x,z)* pairs directly. This "XZ" diagram would stack up many copies of the original diagram, all of the same size, i.e., *without* rescaling them. In this representation, while the picture is always constant size, the viewing window must grow and shrink as it moves up and down in *z*, indicating its changing scope as it zooms. Thus while the world representation is simple, the viewer behavior is complex. In contrast, the "UV" representation of the space-scale diagrams focussed on in this paper can be considered view-based. Conceptually, the world is statically prescaled, and the window is rigidly moved about. The UV representation is thus very useful in discussing how the views should behave. The coordinate transform formulas allow problems stated and solved in terms of view behavior, i.e., in the UV domain, to have their solutions transformed back into XZ for implementation.

EXAMPLE USES OF SPACE-SCALE DIAGRAMS

With these preliminaries, we are prepared to consider various uses of space-scale diagrams. We begin with a few examples involving navigation in zoomable interfaces, then consider how the diagrams can help visualize multiscale objects, and finish by showing how other, non-zoom multiscale views can be characterized.

Pan-zoom trajectories

One of the dominant interface modes for looking at large 2D worlds is to provide an undistorted window onto the world, and allow the user to pan and zoom. This is used in [2][3][7][10], as well as essentially all map viewers in GISs (geographic information systems). Space-scale diagrams are a very useful way for researchers studying interfaces to visualize such interactions, since moving a viewing window around via pans and zooms corresponds to taking it on a trajectory through scale-space. If we represent the window by its midpoint, the trajectories become curves and are easily visualized in the space-scale diagram. In this section, we first show how easily space-scale diagrams represent pan/zoom sequences. Then we show how they can be used to solve a very concrete interface problem. Finally we analyze a more sophisticated pan/zoom problem, with a rather surprising information theoretic twist.

Basic trajectories. Figure 6 shows how the basic pan-zoom trajectories can be visualized. In a simple pan (a) the window's center traces out a horizontal line as it slides through space at a fixed scale. A pure zoom around the center of the window follows a great ray (b), as the window's viewing scale changes but its position is constant. In a "zoom-around" the zoom is centered around some fixed point other than the center of the window, e.g., *q* at the right hand edge of the window. Then the trajectory is a straight line parallel to the great ray of that fixed point. This moves the window so that the

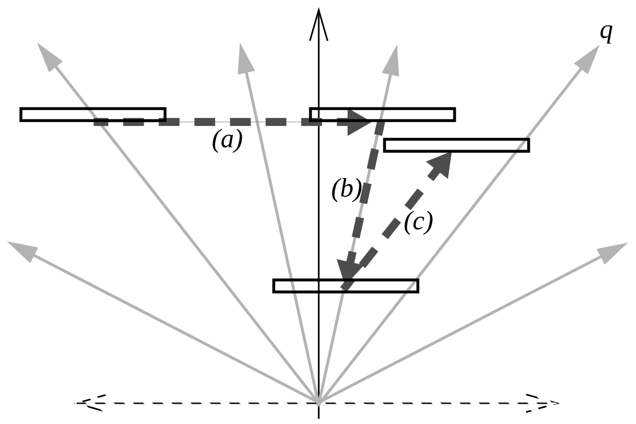

Figure 6. *Basic Pan-Zoom trajectories are shown in the heavy dashed lines:. (a) Is a pure Pan,. (b) is a pure Zoom (out), (c) is a "Zoom-around" the point q.*

fixed point stays put in the view. In the figure, for example, the point, *q*, always intersects the windows on trajectory (c) at the far right edge, meaning that the point, *q*, is always at that position in the view. If as in this case the fixed point is itself within the window, we call it a zoom-around-within-window or *zaww*. Other sorts of pan-zoom trajectories have their characteristic shapes as well and are hence easily visualized with space-scale diagrams.

The joint pan-zoom problem. There are times when the system must automatically pan and zoom from one place to another, e.g. moving the view to show the result of a search. Making a reasonable joint pan and zoom is not entirely trivial. The problem arises because in typical implementations, pan is linear at any given scale, but zoom is logarithmic, changing magnification by a constant factor in a constant time. These two effects interact. For example, suppose the system needs to move the view from some first point (x_1, z_1) to a second point (x_2, z_2). For example, a GIS might want to shift a view of a map from showing the state of Kansas, to showing a zoomed in view of the city of Chicago, some thousand miles away. A naive implementation might compute the linear pans and log-linear zooms separately and execute them in parallel. The problem is that when zooming in, the world view expands exponentially fast, and the target point x_2 runs away faster than the pan can keep up with it. The net result is that the target is approached non-monotonically: it first moves away as the zoom dominates, and only later comes back to the center of the view. Various seat-of-the pants guesses (taking logs of things here and there) do not work either.

What is needed is a way to express the desired monotonicity of the view's movement in both space and scale. This view-based constraint is quite naturally expressed in the UV space-scale diagram as a bounding parallelogram (Figure 7). Three sides of the parallelogram are simple to understand. Since moving up in the diagram corresponds to increasing magnification, any trajectory which exits the top of the parallelogram would have overshot the zoom-in. A trajectory exiting the bottom would have zoomed out when it should have been

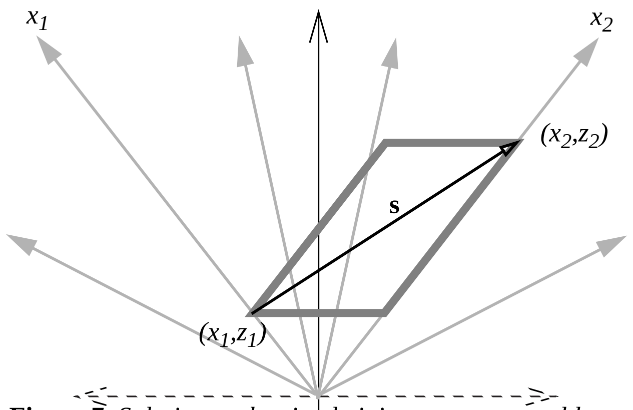

Figure 7. *Solution to the simple joint pan-zoom problem. The trajectory* **s** *monotonically approaches point 2 in both pan and zoom.*

zooming in. One exiting the right side would have overshot the target in space. The fourth side, on the left, is the most interesting. Any point to the left of that line corresponds to a view in which the target x_2 is further away from the center of the window than where it started, i.e., violating the non-monotonic approach. Thus any admissible trajectory must stay within this parallelogram, and in general must never move back closer to this left side once it has moved right. The simplest such trajectory in UV space is the diagonal of the parallelogram. Calculating it is simple analytic geometry. The coordinates of points 1 and 2 would typically come from the implementation in terms of XZ. These would first be transformed to UV. The linear interpolation is done trivially there, and the resulting equation transformed back to XZ for use in the implementation. If one composes all these algebraic steps into one formula, the trajectory in XZ for this 1-D case is:

$$z = \frac{z_1 - m\, z_1 x_1}{1 - m\, x} \qquad \textit{where} \qquad m = \frac{z_2 - z_1}{z_2 x_2 - z_1 x_1}$$

Thus to get a monotonic approach, the scale factor, *z*, must change hyperbolically with the panning of *x*. This mathematical relationship is not easily guessed but falls directly out of the analysis of the space-scale diagram. We implemented the 2D analog in Pad++ and found the net effect is visually much more pleasing than our naive attempts, and count this as a success of space-scale diagrams.

Optimal pan-zooms and shortest paths in scale-space. Since panning and zooming are the dominant navigational motion of these undistorted multiscale interfaces, finding "good" versions of such motions is important.The previous example concerned finding a trajectory where "good" was defined by monotonicity properties. Here we explore another notion of a "good" trajectory, where "good" means "short".

Paradoxically, in scale-space the shortest path between two points is usually not a straight line. This is in fact one of the great advantages of zoomable interfaces for navigation, and results from the fact that zoom provides a kind of exponential accelerator for moving around a very large space. A vast distance may be traversed by first zooming out to a scale where

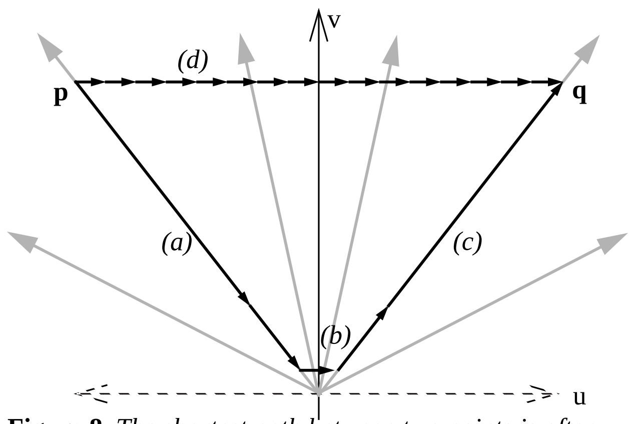

Figure 8. *The shortest path between two points is often not a straight line. Here each arrow represents one unit of cost. Because zoom is logarithmic, it is often "shorter" to zoom out (a), make a small pan (b), and zoom back in (c), than to make a large pan directly (d).*

the old position and new target destination are close together, then making a small pan from one to the other, and finally zooming back in (see Figure 8). Since zoom is naturally logarithmic, the vast separation can be shrunk much faster than it can be directly traversed, with exponential savings in the limit. Such insights raise the question of what is really the optimal shortest path in scale-space between two points.

When we began pondering this question, we noted a few important but seemingly unrelated pieces of the puzzle. First, one naive intuition about how to pan and zoom to cross large distances says to zoom out until both the old and new location are in the view, then zoom back into the new one. Is this related at all to any notion of a shortest path? Second, window size matters in this intuitive strategy: if the window is bigger, then you do not have to zoom out as far to include both the old and new points. A third piece of the puzzle arises when we note that the "cost" of various pan and zoom operations must be specified formally before we can try to solve the shortest path question. While it seems intuitive that the cost of a pure pan should be linear in the distance panned, and the cost of a pure zoom should be logarithmic with change of scale, there would seem to be a puzzling free parameter relating these two, i.e. telling how much pan is worth how much zoom.

Surprisingly, there turns out to be a very natural information metric on pan/zoom costs which fits these pieces together. It not only yields the linear pan and log zoom costs, but also defines the constant relating them and is sensitive to window size. The metric is motivated by a notion of visual informational simplicity: the number of bits it would take to efficiently transmit a movie of a window following the trajectory.

Consider a movie made of a pan/zoom sequence over some 2D world. Successive frames differ from one another only slightly, so that a much more efficient encoding is possible. For example if successive frames are related by a small pan operation, it is necessary only to transmit the bits corresponding to the new pixels appearing at the leading edge of the panning window. The bits at the trailing edge are thrown away. The 1-D version is shown in Figure 9a. If the bit density is β (i.e., bits per centimeter of window real estate), then the number of bits to transmit a pan of size d is $d\beta$.

Similarly, consider when successive frames are related by a small pure zoom-in operation (Figure 9b), say where a window is going to magnify a portion covering only $(w\text{-}d)/w$ of what it used to cover (where w is the window size). Then too, $d\beta$ bits are involved. These are the bits thrown away at the edges of the window as the zoom-in narrows its scope. Since this new smaller area is to be shown magnified, i.e., with higher resolution, it is exactly this number of bits, $d\beta$, of high resolution information that must be transmitted to augment the lower resolution information that was already available.

The actual calculation of information cost for zooms requires a little more effort, since the amount of information required to make a total zoom by a factor r depends on the number and size of the intermediate steps. For example, two discrete step zooms by a factor of 2 magnification require more bits than a single step zoom by a factor of 4. (Intuitively, this is because showing the intermediate step requires temporarily having some new high resolution information at the edges of the window that is then thrown away in the final scope of the zoomed-in window.) Thus the natural case to consider is the continuous limit, where the step-size goes to zero. The resulting formula says that transmitting a zoom-in (or out) operation for a total magnification change of a factor r requires $\beta w \log(r)$ bits.

Thus the information metric, based on a notion of bits required to encode a movie efficiently, yields exactly what was promised: linear cost of pans ($d\beta$), log costs of zooms ($\beta w \log(r)$), and a constant (w) relating them that is exactly the window size. Similar analyses give the costs for other ele-

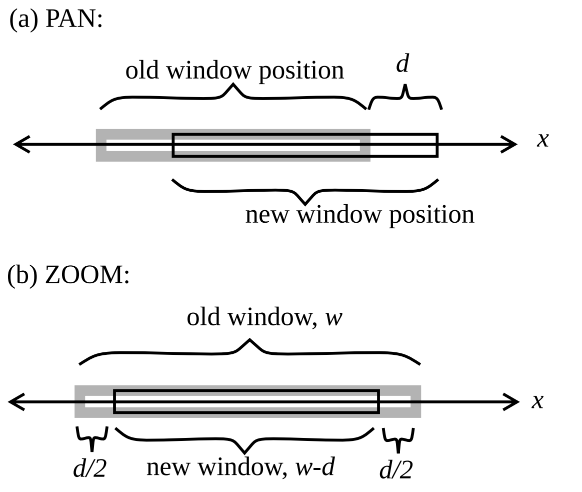

Figure 9. *Information metric on pan and zoom operations on a 1-D world. (a) Shifting a window by d requires d*β *new bits. (b) Zooming in by a factor of (w-d)/w, throws away d*β *bits, which must be replaced with just that amount of diffuse, higher resolution information when the window is magnified and brought back to full resolution.*

mentary motions. For example, a zoom around other any other point within the window (a zaww) always turns out to have the same cost as a pure (centered) zoom. Other arbitrary zoom-arounds are a bit more complicated.

From these components it is possible to compute the costs of arbitrary trajectories, and therefore in principle to find minimal ones. Unfortunately, the truly optimal ones will have a complicated curved shape, and finding it is a complicated calculus-of-variations problem. We have limited our work so far to finding the shortest paths within certain parameterized families of trajectories, all of which are piecewise pure pans, pure zooms or pure zaww's. We sketch typical members of the families on a space-scale diagram, pick parameterizations of them and apply simple calculus to get the minimal cases. There is not room here to go through these in detail, but we give an overview of the results.

Before doing so, however, it should be mentioned that, despite all this formal work, the real interface issue of what constitutes a "good" pan/zoom trajectory is an empirical/cognitive one. The goal here is to develop a candidate theory for suggesting trajectories, and possibly for modelling and understanding future empirical work. The suitability of the information based approached followed here hinges on an implicit cognitive theory that humans watching a pan/zoom sequence have somehow to take in, i.e., encode or understand, the sequence of views that is going by. They need to do this to interpret the meaning of specific things they are seeing, understand where they are moving to, how to get back, etc. It is assumed that, other things being equal, "short" movies are somehow easier, taking fewer cognitive resources (processing, memory, etc.) than longer ones. It is also assumed that human viewers do not encode successive frames of the movie but that a small pan or small zoom can be encoded as such, with only the deltas, i.e., the new information, encoded. Thus to some approximation, movies with shorter encoded lengths will be better. (We are also at this point ignoring the content of the world, assuming that no special content-based encoding is practical or at least that information density at all places and scales is sufficiently uniform that its encoding would not change the relative costs.)

To get some empirical idea of whether this information-theoretic approach to "goodness" of pan-zoom trajectories matches human judgement, we implemented some simple testing facilities. The testing interface allows us to animate between two specified points (and zooms) with various trajectories, trajectories that were analyzed and programmed using space-scale diagrams. We did informal testing among a few people in our lab to see if there was an obvious preference between trajectories, and compared these to the theory.

For large separations, pure pan is very bad. There is strong agreement between theory and subjects' experience. Theory says the information description of a pure pan movie should be exponentially longer one substantially using a zoom. Empirically, users universally disliked these big pans. They found it difficult to maintain context as the animation flew across a large scene. Further, when the distance to be travelled was quite large and the animation was fast, it was hard to see what was happening; if the animation was too slow, it took too long to get there.

At the other extreme, for small separations viewers preferred a short pure pan to strategies that zoomed out and in. It turns out that this is also predicted by the theory for the family piecewise pan/zoom/zaww trajectories we considered here. Depending on exactly which types of motions are allowed, the theory predicts that to traverse separations of less than 1 to 3 window widths, the corresponding movie is informationally shorter if it is just a pan.

Does the naive zoom-pan-zoom approach ever obtain? Well at the level of zoom out till the two are close, then pan in, the answer is definitely yes. The fine points are a bit more subtle. If only zaww's are allowed, the shortest path indeed involves zooming out until both are visible then zooming in (Figure 10). For users this was quite a well-liked trajectory. If pans

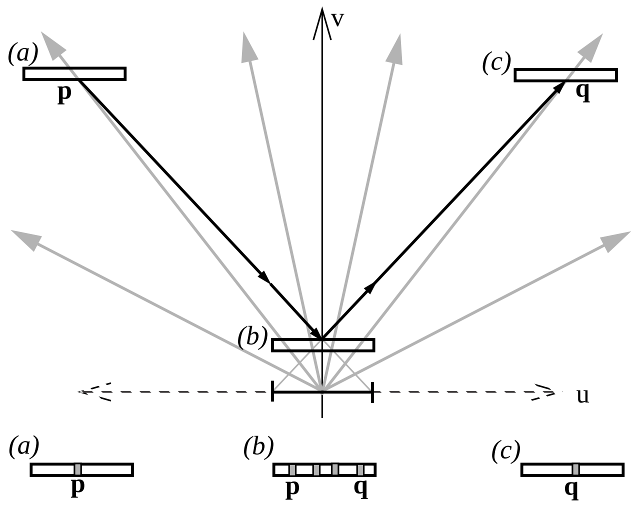

Figure 10. *The shortest zaww path between* **p** *(a) and* **q** *zooms out till both are within the window (b), then zooms in (c). The corresponding views are shown below the diagram.*

are allowed, however, the information metric disagrees slightly with the naive intuition, and says to stop the zoom before both are in view, and make a pan of 1-3 screen separations (just as described for short pans), then zoom in. The information difference between this optimal strategy and the naive one is small, and our users similarly found small differences in the acceptability. It will be interesting to examine these variants more systematically.

Our overall conclusion is that the information metric, based on analyses of space-scale diagrams, is quite a reasonable way to determine "good" pan/zoom trajectories.

Showing semantic zooming

Another whole class of uses for space-scale diagrams is for the representation of *semantic zooming*[10]. In contrast to geometric zooming, where objects change only their size and not their shape when magnified, semantic zooming allows objects to change their appearance as the amount of real-estate available to them changes. For example, an object could just appear as a point when small. As it grows, it could then in turn appear as a solid rectangle, then a labeled rectangle, then a page of text, etc.

Figure 11 shows how geometric zooming and semantic zooming appear in a space-scale diagram. The object on the left, shown as an infinitely extending triangle, corresponds to a 1-D gray line segment, which just appears larger as one zooms in (upward: 1,2,3). On the right is an object that changes its appearance as one zooms in. If one zooms out too far (a), it is not visible. At some transition point in scale, it suddenly appears as a three segment dashed line (b), then as a solid line (c), and then when it would be bigger than the window (d), it disappears again.

The importance of such a diagram is that it allow one to see several critical aspects of semantic objects that are not otherwise easily seen. The transition points, i.e., when the object changes representation as a function of scale, is readily apparent. Also the nature of the changing representations, what it looks like before and after the change, can be made clear. The diagram also allows one to compare the transition points and representations of the different objects inhabiting a multi-scale world.

We are exploring direct manipulation in space-scale diagrams as an interface for multi-scale authoring of semantically zoomable objects. For example, by grabbing and manipulating transition boundaries, one can change when an object will zoom semantically. Similarly, suites of objects can have their transitions coordinated by operations analogous to the snap, alignment, and distribute operators familiar to drawing programs, but applied in the space-scale representation.

As another example of semantic zooming, we have also used space-scale diagrams to implement a "fractal grid." Since grids are useful for aiding authoring and navigation, we wanted to design one that worked at all scales -- a kind of virtual graph paper over the world, where an ever finer mesh of squares appears as you zoom in. We devised the implementation by first designing the 1D version using the space-scale diagram of Figure 12. This is the analog of a ruler where ever finer subdivisions appear, but by design here they appear only when you zoom in (move upward in the figure). There are nicely spaced gridpoints in the window at all five zooms of the figure. Without this fractal property, at some magnification, the grid points would disappear from most views.

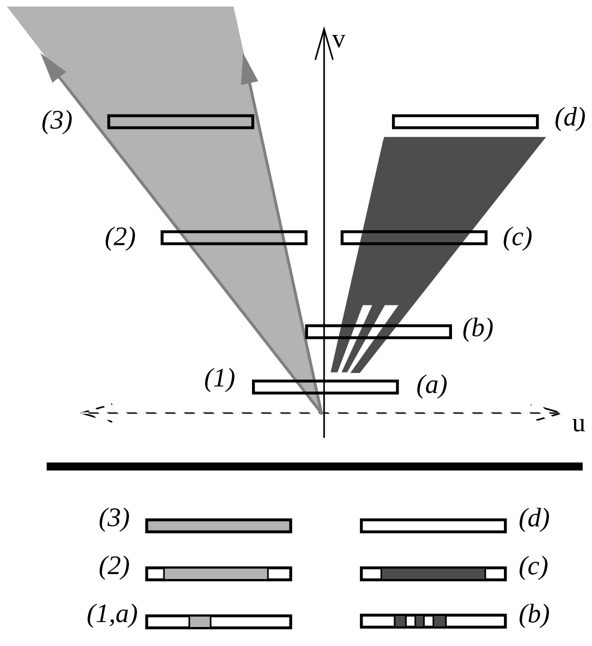

Figure 11. *Semantic Zooming. Bottom slices show views at different points.*

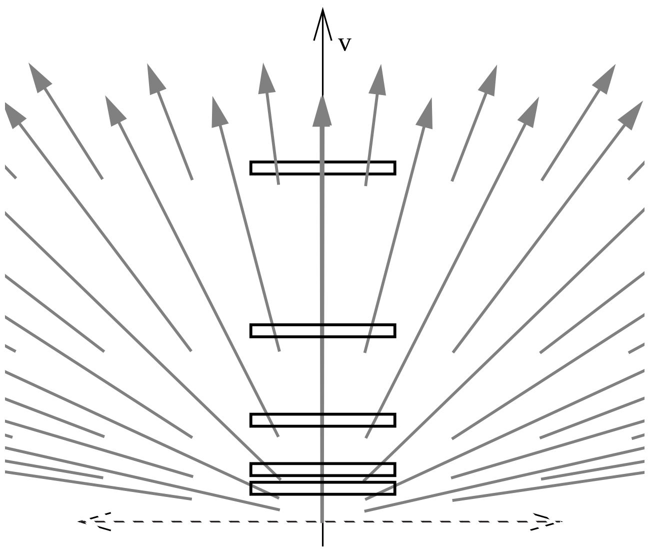

Figure 12. *Fractal grid in 1D. As the window moves up by a factor of 2 magnification, new gridpoints appear to subdivide the world appropriately at that scale. The view of the grid is the same in all five windows.*

Warps and fisheye views

Space-scale diagrams can also be used to produce many kinds of image warpings. We have characterized the space-scale diagram as a stack of image snapshots at different zooms. So far in this paper, we have always taken each image as a horizontal slice through scale space. Now, instead imagine taking a cut of arbitrary shape through scale space and projecting down to the *u* axis. Figure 13 shows a step-up-step-down cut that produces a mapping with two levels of magnification and a sharp transition between them. Here, *(a)* shows the trajectory through scale space, *(b)* shows the result that would obtain if the cut was purely flat at the initial level, and (c) shows the warped result following.

Different curves can produce many different kinds of mappings. For example, Figure 14 shows how we can create a fisheye view.* By taking a curved trajectory through scale-space, we get a smooth distortion that is magnified in the center and compressed in the periphery. Other cuts can create bifocal [1] and perspective wall [8].

* In fact exactly this strategy for creating 2D fisheye views was proposed years ago in [5], p 9,10.

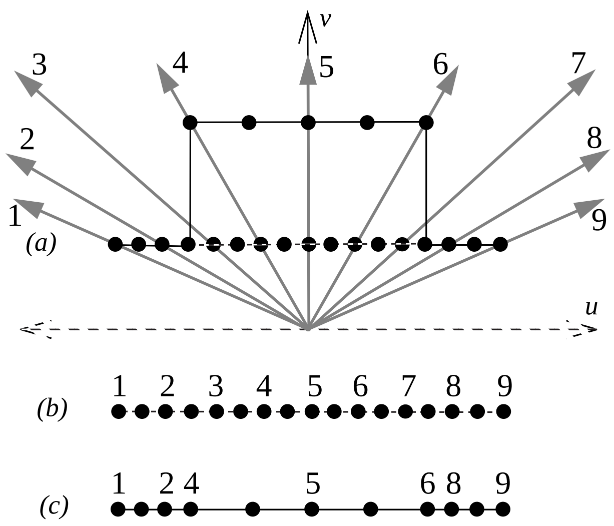

Figure 13. *Warp with two levels of magnification and an abrupt transition between them. (a) shows the trajectory through scale-space, (b) shows the unwarped view, and (c) shows the warped view (notice rays 3 and 7 don't appear).*

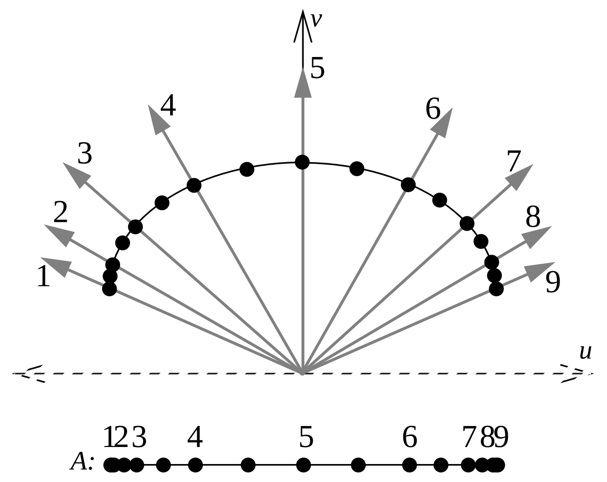

Figure 14. *Fisheye view.*

For cuts as in Figure 13, which are piece-wise horizontal, the magnification of the mapping comes directly from the height of the slice. When the cuts are curved and slanted, the geometry is more complicated but the magnification can always be determined by looking at the projection as in Figure 14. [*]

CONCLUSION

This paper introduces space-scale diagrams as a new technique for understanding multiscale interfaces. Their defining characteristic and principal virtue is that they represent scale explicitly. We showed how they can aid the analysis of pans and zooms because they take a temporal structure and turn it into a static one: a sequence of views becomes a curve in scale-space. This has already helped in the design of good pan/zoom trajectories for Pad++. We showed how the diagrams can help visualization of semantic zooming by showing an object in all its scale-dependent versions simultaneously. We expect to use this as an interface for designing semantically zoomable objects. We also suggested that diagrams may be useful for examining other non-flat multiscale representation, such as fisheye views.

Space-scale diagrams, therefore, are important for visualizing various problems of scale, for aiding formal analyses of those problems, and finally, for implementing various solutions to them.

* Simple projection is only one way for such cuts to create views. For example if one takes the XZ transformed version of these diagrams with cuts, one can use them directly as the magnification functions of [6].

REFERENCES

1. Apperley, M.D., Tzavaras, I. and Spence, R, A Bifocal Display Technique for Data Presentation, Proceedings of Eurographics '82, pp. 27-43.
2. Bederson, B. B. and Hollan, J.D., Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics, Proceedings of ACM UIST'94, (1994, Marina Del Ray, CA), ACM Press.
3. Donelson, W., Spatial Management of Information, SigGraph'78 (Atlanta, GA), pp. 203-209.
4. Furnas, G.W., Generalized Fisheye Views. In Proceedings of CHI'86 Human Factors in Computing Systems (Boston, MA, April 1986), ACM press, pp. 16-23
5. Furnas, G. W., The FISHEYE view: A new look at structured files.Bell Laboratories Technical Memorandum, #82-11221-22, Oct 18, 1982. 22pps.
6. Leung, Y.K. and Apperley, M.D., A Unified Theory of Distortion-Oriented Presentation Techniques. In press, TOCHI.
7. Lieberman, H., Powers of Ten Thousand: Navigating in Large Information Spaces. Short paper, upcoming UIST'94.
8. Mackinlay, J.D., Robertson, G.G. and Card, S.K., The Perspective Wall: Detail and Context Smoothly Integrated, ACM, Proc. CHI'91, pp. 173-179.
9. Noik, E.G., A Space of Presentation Emphasis Techniques for Visualizing Graphs, GI '94: Graphics Interface 1994, (Banff, Alberta, Canada, May 16-20, 1994), pp. 225-234.
10. Perlin, K. and Fox, D., Pad: An Alternative Approach to the Computer Interface, SigGraph `93 (Anaheim, CA) pp. 57-64
11. Robertson, George. G. and Mackinlay, Jock, The Document Lens, in Proc. UIST'93 (Atlanta, GA) ACM, pp. 101-108.
12. Sarkar, M. and Brown, M.H. Graphical fisheye views of graphs. In Proc. ACM CHI'92, pages 83-91, Monterey, CA, May, 1992. ACM.
13. Sarkar, M., Snibbe, S.S. and Reiss, S.P. Stretching the Rubber Sheet: A Metaphor For Visualizing Large Structures on Small Screens. In ACM UIST '93 (November 1993), ACM Press.