Critical Zones in Desert Fog: Aids to Multiscale Navigation

Susanne Jul
Computer Science and Engineering
University of Michigan
1101 Beal Av.
Ann Arbor MI 48109-2110 USA
+1 734-213-2720
sjul@umich.edu

George W. Furnas
School of Information
University of Michigan
3080 West Hall
Ann Arbor MI 48109-1092 USA
+1 734-763-0076
furnas@umich.edu

ABSTRACT
In this paper, we introduce the problem of *"desert fog,"* a condition wherein a view of an information world contains no information on which to base navigational decisions. We present a set of view-based navigational aids that allow navigators to find their way through desert fog in multiscale electronic worlds. Prototypes of these aids have been implemented in the *Landmarking* and *ZTracker* systems. We introduce the concept of *critical zone analysis,* a method of grouping objects according to their visibility in views of the information world rather than their spatial layout. This concept was derived from a formal analysis of desert fog using view-navigation theory. Our analysis informally extends view-navigation theory to accommodate spatial multiscale worlds and is detailed in the paper.

KEYWORDS
Navigation, Browsing, Information Navigation, Multiscale, Residue, View-Navigation, View-Based Navigational Aids, Critical Zones, Critical Zone Analysis, Pad++, Space-Scale Diagrams, ZTracker.

INTRODUCTION

> *The unwavering fog limited her vision to a small rectangle of utterly uniform sand. The Oasis had been to her right earlier, but the sudden fog only let her guess at where it was now. She couldn't even tell if she had moved up or down. With mounting desperation, she rose a bit. The next rectangle of sand bounded by fog was identical to the last rectangle of sand bounded by fog.*
>
> SJul, *Visions of Desert Fog*

> *I've been lost before; this is exactly what it looks like.*
>
> Hawkeye Pierce, *M*A*S*H*

Many human endeavors entail navigation through an environment. Interaction with electronic worlds is no exception. However, electronic worlds often present new navigational problems as well as new ways of solving them. In the scenario just described, the navigator is faced with a situation where the immediate environment is totally devoid of navigational cues. This is a rare occurrence in the physical world, but is common and inherent in certain types of electronic worlds. We have called this phenomenon *"desert fog,"* and seek to understand it as well as to find ways of helping users to find their way through it.

Desert fog can occur in many electronic environments, but is inherent and pervasive in *multiscale* worlds. These are worlds in which information can exist at multiple levels of detail—from microscopic features to major landscapes. In our experience with Pad++ [1], a widely-distributed experimental multiscale environment, desert fog causes severe navigational problems for both novice and experienced users. With infinite scale, there is always more room between things to explore. Like Hawkeye, navigators in Pad++ know what being lost looks like: Nothing.

By *navigation* we mean the cognitive process of determining and following a path, based on knowledge of and information in the environment. The subordinate tasks of *traversal*—movement through the environment—and *steering*—controlling movement—are assumed in this definition [7] and are often the subjects of research on navigation. The focus of our work is on characterizing and supplying the information needed by the navigator to make correct navigational decisions. We work in the context of multiscale worlds, but focus especially on *spatial multiscale* worlds. These combine multiscale with strongly physical metaphors for interaction, such as fly-through or zooming, and are becoming increasingly common.

Our goal is to design ways of helping users to navigate through very large, constantly changing multiscale information worlds that contain desert fog. In this paper, we present a navigational aid that is based on the concepts of *critical zones* and *critical zone analysis.* Critical zone analysis dynamically identifies groups of objects that are defined by what objects may appear together in views of the world. It is applied to Pad++ in the *ZTracker* prototype system. We derive our approach from an analysis of desert

UIST '98. San Francisco, CA

fog using view-navigation theory [3]—a theoretic understanding of necessary navigational properties of information structures.

Throughout our work, we seek solutions that require as few assumptions as possible about the information space, its information content and the navigator. We constrain ourselves by assuming that the information is under full ownership of the user, i.e., that it may not be altered by the system (including reorganizing its spatial layout). In the work reported here, we concentrate on showing the navigator *where* there is information. We leave the equally important problem of describing *what* the information is for the future.

Our work makes two contributions. The first contribution is the concept of critical zone analysis and the navigational aids based thereon. Critical zone analysis provides a means of grouping information that neither assumes nor implies that the spatial layout of the world is meaningful. Groupings are based purely on object visibility. The second contribution is the analysis of desert fog. It provides an understanding of desert fog that can be used in developing a variety of strategies to address it in a variety of contexts. Many of these strategies are applicable to the general class of desert-fog-like problems wherein the environment contains only navigational cues that are not useful to the user.

The descriptions of the design concepts and the theoretical analysis may be read independently of each other. Readers who are familiar with view-navigation theory or who have a particular interest in the details of the theoretical analysis may wish to read the section "View-Navigation Analysis" before the section "Addressing Desert Fog."

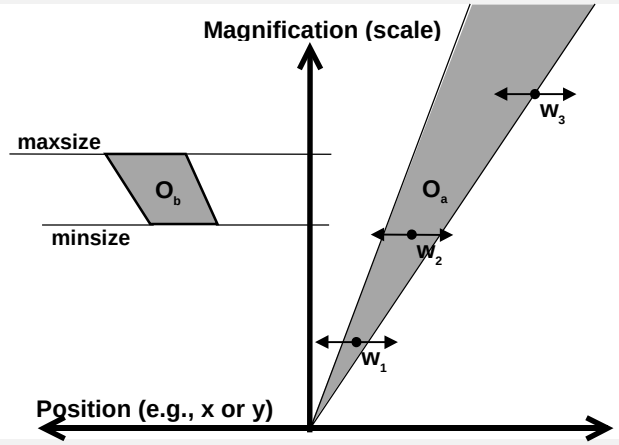
RELATED WORK

Desert fog has become a more pressing concern with the emergence of spatial multiscale interfaces but is not a new problem. Numerous designers have devised ways of managing desert fog problems. Typically, they address the problem either when the space itself is created or when information is placed within it.

Under strategies that address the problem when the space is created, there is, in the beginning, nothing, not even empty space. Authors have to create everything, including patches of desert fog. Many authoring tools, such as the *Rooms* environment [5], and hypertext systems rely upon such strategies to prevent desert fog. Few authors would create a maze of identical rooms or identical web pages.

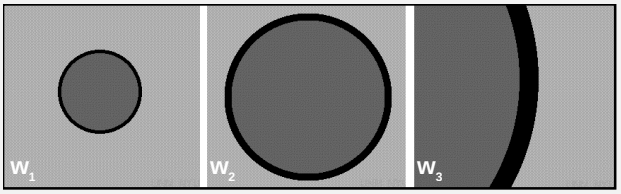
Under strategies that address the problem when the information is embedded in the space, there is, in the beginning, only empty space which the author is seeking to populate. These strategies are generally enforced only by designer and implementor discipline. The common approach is to create an island of information surrounded by desert fog and make it clear that there be desert fog beyond the boundaries of the island. This strategy is

UNDERSTANDING SPACE-SCALE DIAGRAMS



Space-scale diagrams were developed as a tool for understanding multiscale spaces [4]. They show the apparent change in size and position of an object relative to the magnification of the view. In the sample diagram above, the horizontal axis indicates location in screen-space (e.g., x-coordinate) and the vertical axis indicates degree of magnification (the scale-coordinate). Note that zooming "in" and "out" correspond to moving "up" and "down," respectively, in a space-scale diagram.

In the simple case, an object only grows in size as it is magnified. Such geometrically-scaling objects, like $\mathbf{O}_a$ in the sample diagram, have a V shape in a space-scale diagram, indicating that the object appears to be infinitely small at infinitely small scales, and keeps growing larger as the view is magnified. In real interfaces, an object typically has a minimum visible size, its *minsize*, and disappears, at least, when it is smaller than a pixel. Objects also have a maximum effective size, the *maxsize;* e.g., when they fill the view uniformly they are often culled by the rendering system. These limits are shown schematically for object $\mathbf{O}_b$ in the sample diagram.



A particular view of the world is defined by the position in space and scale of a window with a given width. This is represented in a space-scale diagram by a horizontal line whose midpoint represents of the center of the window. (Note that we assume uniform magnification across any particular view.) Since the width of the window is unaffected by the magnification of the view, a line representing a particular window will have the same width throughout the diagram. In the sample diagram, $\mathbf{w}_1$ is a view in which $\mathbf{O}_a$ fills the middle third of the window, as shown in the first of the screen-shots above. $\mathbf{w}_2$ has zoomed in on (the now magnified) $\mathbf{O}_a$, as shown in the second screen-shot. $\mathbf{w}_3$ has zoomed in further and panned right almost half a window width, as shown in the third screen-shot.

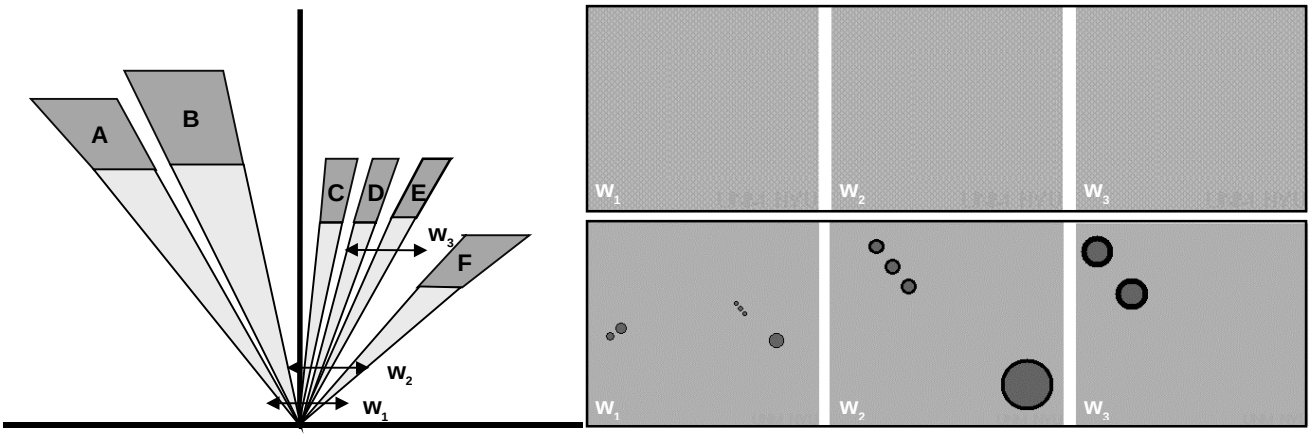


Figure 1 A multiscale world with no navigational aids and six objects, **A-F**. Without navigational aids most views in the world are in desert fog—including $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ as can be seen from the screen-shots in the right top row.

This figure (and several of the following) is composed of three parts. (1) On the left, a space-scale diagram containing three views, $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$. (Note that we only provide "x–z" space-scale diagrams that do not account for the y-coordinate.) (2) On the right, top row, three screen-shots from Pad++ corresponding to the views from $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$. (3) On the right, bottom row, the views from $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ as they would appear *if they had no minsize (as indicated by the lightly shaded "V"s in the diagram).* For the purposes of illustration, the relative sizes of the objects are designed so that there are views in which all objects could be seen (as in $\mathbf{w}_1$) if their minsizes permitted. In normal use, it would be highly likely that there would be no view in which all objects, or even objects **C-E** and object **F** (as in $\mathbf{w}_2$), could appear together. This distortion underplays the severity of the desert fog problem.

typically applied to homogeneous cohesive collections of information that readily form single or well-connected islands of information. The *WING* [11] interactive tourist guide and *TimeLines* [10] visualization of the history of photography are typical examples of this strategy.

General solutions can be imagined that address desert fog at either space-creation or information-embedding time. However, we have chosen to address desert fog at navigation time. We assume that the world contains desert fog and the navigator needs to find their way through it.

DESERT FOG IN PAD++

Pad++ provides a concrete context for our work and has served as the testbed for our ideas. Before embarking on a solution to desert fog, we describe Pad++ [1] and how desert fog affects interactions in Pad++ worlds. In this and following discussions, we rely heavily upon the space-scale notation developed by Furnas and Bederson [4], and described in the box "Understanding Space-Scale Diagrams."

The metaphor for Pad++ is an infinite two-dimensional surface that can be magnified uniformly and infinitely. Interaction is through panning and zooming. Objects have position and extent on the surface, but appear differently on the screen depending on the magnification or scale of the current view.

There are two causes of desert fog in Pad. First, desert fog arises because the space (the Pad++ surface) exists independently of any embedded information and is itself visually homogeneous. In a Pad++ world with no embedded information, there are no navigational clues anywhere. With information embedded, there are navigational clues only in views in which information objects can actually be seen. Second, desert fog arises because objects have a natural or imposed minimum (and, possibly, maximum) size at which they are rendered. That allows information objects to be invisible in some views even though they are contained in those views. (By *contained*, we mean that the view can be reached simply by zooming in.) Any solution to the desert fog problem must speak to both sources.

Figure 1 illustrates a multiscale world with six objects and much desert fog. A navigator has several choices in any view. They can zoom out (decrease the magnification—"down" in a space-scale diagram), zoom in (increase the magnification—"up" in a space-scale diagram), or pan in a variety of directions. Since there is no way to tell which is the correct choice in a desert fog view, it is not surprising that even experienced Pad++ users laugh when asked to accomplish tasks that require navigation through desert fog.

ADDRESSING DESERT FOG

We include the full view-navigation analysis of desert fog later, but summarize the results here to motivate the development of our navigational aids.

The analysis revealed two properties that must hold for a general solution to desert fog. First, there must be a single unambiguous action to be taken in a desert fog view. This is consonant with the intuition that the problem in desert fog is not that there is no navigational information, but rather that the navigator does not know what to do when there is no navigational information. Second, all navigational information provided must be structured so that only a

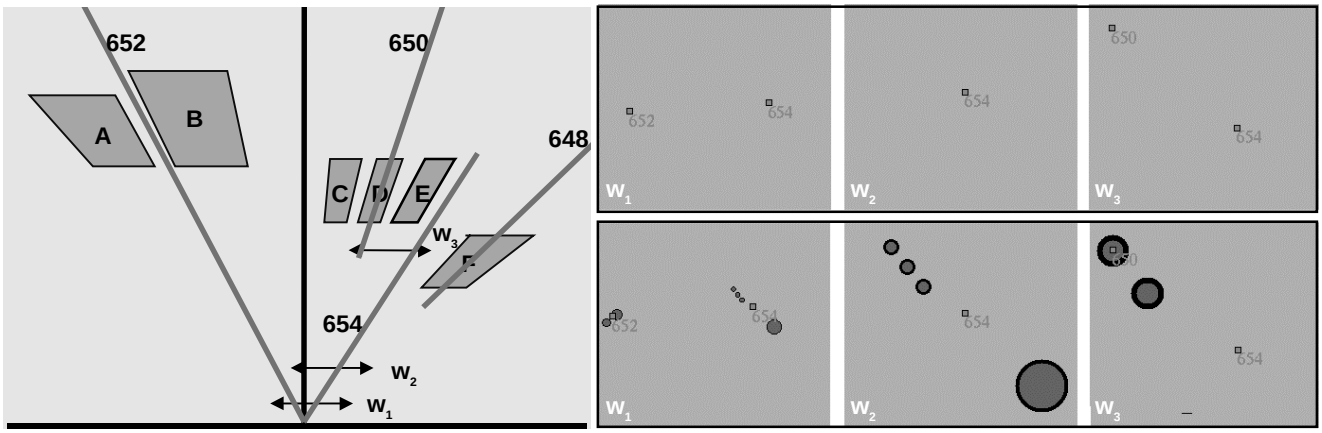


Figure 2 Multiscale hierarchical residue in the Landmarking system. Objects **A** and **B** form a cluster marked with landmark **Q**. Objects **C-E** form a cluster marked with landmark **R**. Object **F** is in a cluster of its own, marked with landmark **S**. The clusters marked by **R** and **S**, in turn, form a composite cluster, marked with landmark **T**. The visibility range of a landmark is, in this version, determined as being from the maxsize of its associated cluster, if it is a composite cluster, to the minsize of its containing cluster if is a member of a composite cluster.

Landmarks appear as small red squares (resembling conventional grab-handles). At this time, there is no visual distinction between landmarks marking simple and composite clusters. The right upper row of images is screen-shots from the Landmarking system and the lower row shows the views as they would appear if the objects were made visible. As the user zooms from $\mathbf{w}_1$ toward $\mathbf{w}_2$, the landmark for cluster **652** (labeled in the screen-shot) will move left, eventually, out of the view, leaving the landmark for cluster **654** in the view. As the user zooms from $\mathbf{w}_2$ toward $\mathbf{w}_3$, the landmark for cluster **650** eventually appears or moves into the view from the left.

small amount is presented in any given view. This is consonant with the intuition that too much information will clutter the view and, potentially, confuse the navigator.

To provide a single unambiguous action, we create *multiscale residue* for all objects in a view. *Residue*, a view-navigation term, is evidence that leads a navigator to believe that a particular object may be found in a particular direction. Multiscale residue is residue that exists across scale. If an object has multiscale residue, this residue will be visible in all views in which the object is contained. The object can then be reached by zooming in. If the object does not have residue in a particular view, the object is not contained in that view. Zooming out will—due to its converging property—always lead to a view in which the object's residue appears. If there is no residue of an object in one view, it can be reached by zooming out until its residue is visible, then zooming in on the residue. We experimented with two ways of providing multiscale residue. The first indicates particular points in space, while the second indicates particular regions.

To reduce the amount of navigational information in each view, we experimented with two ways of grouping objects so that they can share residue. The first is based on traditional cluster analysis and the second is based on the visibility of objects in different views.

Cluster Analysis

Our first instinct was to group information objects based on their spatial layout. Assuming that the layout is not random, we applied traditional cluster-analysis techniques [2], using the spatial (x, y) distance between objects as the distance metric.

As an example of this approach, we used single-link agglomerative clustering [2]. Two objects were grouped together if they were within a certain distance of each other. (We experimented with a variety of both fixed and variable distances.) This proximity test was applied, transitively, to all objects in the world, and then, recursively, to the resulting clusters. This yields a nicely hierarchical grouping of the information objects with clusters from a single pass of the algorithm forming a level in the hierarchy. We then added a visual indicator—a *landmark*—to the display for each internal node in the hierarchy. This landmark is independent of scale, so does not grow or shrink as the user zooms in or out (landmarks resemble conventional grab-handles) and, so, constitutes multiscale residue for the group. This results in the space-scale profile and corresponding views from our *Landmarking* system shown in Figure 2.

Two parameters of the display of landmarks proved to be particularly insidious. First, where should the landmark be located? The geometric center of the group is simple, but may not be meaningful in representing the group. It would be more desirable to use the location of the object that, for the user, is most representative of the group. Identifying groups and their representative objects has been investigated by psychologists [8, 9, 12] and was explored in the LEADS system [6]. These efforts all show that it is not a simple matter. The second problematic parameter was how many and which levels of the hierarchy to display at

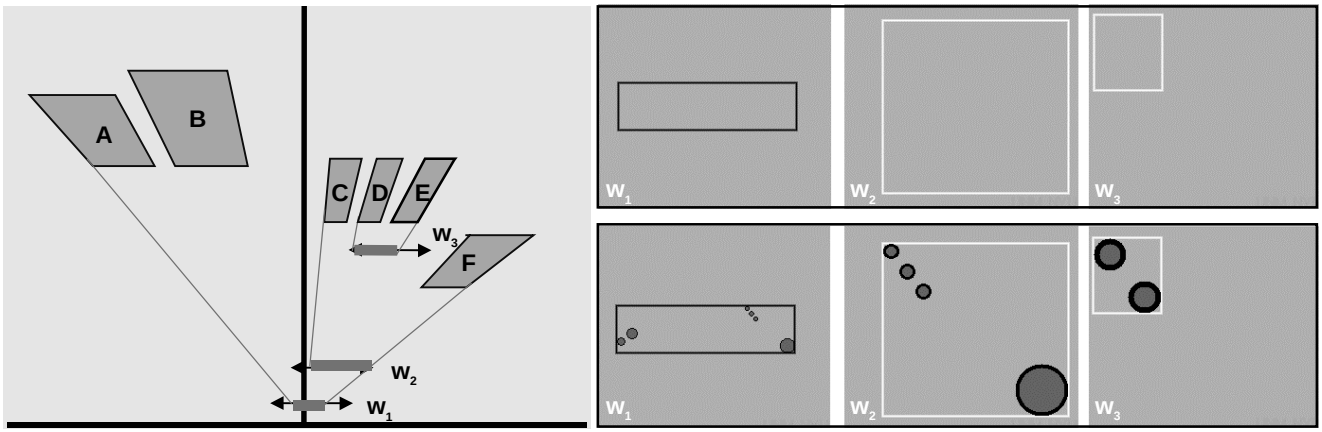


Figure 3 Multiscale residue in ZTracker using the Single Critical Zone algorithm. In the space-scale diagram, the heavily shaded portion of each view ($\mathbf{w_1}$, $\mathbf{w_2}$ and $\mathbf{w_3}$) denotes the critical zone in that view. These critical zones are indicated with rectangles in the screen-shots on the right. The upper row of screen-shots is from the ZTracker system and the lower row is the same views as they would appear if the objects were made visible. The critical zone in $\mathbf{w_1}$ contains objects **A-F**, that of $\mathbf{w_2}$ contains objects **C-F**, and that of $\mathbf{w_3}$ contains **D** and **E**, as can be seen in the lower row of images. The critical zone in $\mathbf{W_1}$ is colored to indicate that it contains all the objects in the world. As the user zooms from $\mathbf{w_1}$ toward $\mathbf{w_2}$ by zooming in on the rectangle in $\mathbf{w_1}$ the rectangle expands. When it reaches the edge of the screen it will disappear and another rectangle (for the critical zone containing **B-F**) will appear. This rectangle, in turn, will grow until it reaches the edge of the screen, when it is replaced by the rectangle in $\mathbf{w_2}$. A similar sequence of transformations occurs as the user zooms in from $\mathbf{w_2}$ toward $\mathbf{w_3}$.

any given time. Clearly, displaying all levels potentially results in cluttered views. Showing a single level at a time, e.g., the lowest level in the view, solves the basic desert fog problem. However, it is still difficult for the navigator to predict where landmarks will appear.

These difficulties highlight a shortcoming of applying cluster analysis to the spatial layout: Semantics and spatial layout are confounded making it necessary to have some understanding of the semantics of the objects in order to provide meaningful navigational guidance. A more insidious problem is the danger of falsely conveying an impression of meaningful structure. The hierarchical structure identified by cluster analysis may or may not have been intended—and therefore be meaningful—by the author(s) of the world, but users may be inclined to assume that it was. Rather than attempting to understand ways of analyzing semantics, we re-visited our view-navigation analysis and realized that "it's about views, not objects."

View Analysis

Upon re-visiting our analysis of desert fog we recognized the significance of views, in general, and of the current view, in particular. Abandoning the effort to provide residue of objects, we sought to provide residue of views. We distinguish between *interesting views* that contain information objects, and desert fog views that do not.

Three key insights underlie our final design approach. First, in order to achieve our goal of helping users to find where there are things in a world containing desert fog, it is sufficient to provide residue only of interesting views. Second, by relying on the unambiguous action provided by multiscale residue, it is sufficient to provide residue only of views that are contained in the current view. Third, it is sufficient for residue to indicate a general direction, as long as this general direction becomes increasingly specific as the view gets closer. A strict requirement is that all interesting views contained in a view must have residue in that view.

Pursuing these insights, we developed the concept of *critical zones*. Intuitively, a critical zone is a region of the view where zooming in leads to interesting views. More formally, a critical zone is a region in a view that is *guaranteed* to contain interesting views. Any one critical zone must be contiguous, and it must be the smallest region of its family of shapes (rectangles, circles, etc.) that contains a particular set of interesting views. This led to the view-based navigational aids that we explored in the *ZTracker* prototyping system.

ZTRACKER

ZTracker is an exploratory system developed in Pad++ to experiment with navigational aids based on critical zones. ZTracker tracks rectangular critical zones as the user moves through the world. In each view, the critical zones (if there are any in that view) are indicated visually by tracing their outlines. Critical zone indicators are multiscale, and grow and shrink like normal objects, but have a fixed minimum size, so never disappear—even at very small scales. As a result, in a view with no critical zones (i.e., a desert fog view), the appropriate navigational strategy is to zoom out until one appears. In the current implementation, critical zone indicators change color when all objects in the world are contained in the current view. This keeps the navigator from zooming out infinitely.

We have developed three algorithms for computing critical zones. One computes the critical zone that contains all

objects in a view, one decomposes this single critical zone into a set of smaller critical zones, and one computes all window-sized critical zones in a world.

Tracking a Single Critical Zone

In the simplest case, we show at most one critical zone. In ZTracker, this critical zone corresponds to the bounding box of all objects contained in the view. More generally, it is the projection onto the window (using the same projection technique as the rendering system) of the bounding volume of all objects contained in the view. It is computed, in ZTracker, simply by asking the Pad++ system to find all objects within the window rectangle, then finding the bounding box of those objects. Figure 3 shows a space-scale diagram and the corresponding screen-shots from ZTracker for the single critical zone method.

Although our experience with tracking a single critical zone is limited, we have found it to be surprisingly helpful. Merely distinguishing desert fog views from interesting views is helpful (and anxiety-reducing). Users quickly become adept at inferring locations of interesting views from changes in critical zones during zooming. One obvious shortcoming of tracking a single critical zone, however, is that it frequently captures a great deal of desert fog. If no objects are actually visible, there is no way for the navigator to predict where in the critical zone to zoom in and they may spend too much time zooming in on desert fog, only to have to backtrack. (Of course, critical zones make it possible to know when to backtrack and how far back to go.)

Tracking Critical Zones Recursively

In order to provide the navigator with more detailed information than a single critical zone, we explored a variety of recursive algorithms. (Note that we still strive to keep the number of critical zones indicated in a view small.)

We use a divide-and-conquer strategy to refine a given critical zone into a set of smaller critical zones. The initial critical zone is always the largest critical zone in the current view (i.e., the critical zone identified by the single-zone algorithm). To refine a critical zone, it is first subdivided into a set of regions—rectangles in ZTracker. (We shall explain how we subdivide a critical zone shortly.) The largest critical zone, if any, within each region is then determined by applying the single-zone algorithm to the region. The algorithm is then applied recursively to each of the resulting critical zones until all critical zones are either smaller than some fixed minimum size or contain only a single object.

On first impression, any means of subdividing critical zones would serve. However, arbitrary divisions such as division into quarters, or dividing along object boundaries, may bisect objects. In the current implementation, only fully contained objects are included in critical zones, so bisected objects would be excluded from any subsequent critical zones and would be lost. This would violate the requirement that all interesting views have residue in views in which they are contained.

The subdividing algorithm implemented in ZTracker uses the window pixellation to determine subdivisions. (This is, in some ways, similar to ray-tracing algorithms used in computer graphics). A (rectangular) critical zone is divided into four regions by shrinking it on each side successively by one pixel-width. (One pixel-width may correspond to many units on the Pad++ surface, so many objects may be eliminated at once.) This results in four over-lapping regions, as illustrated in Figure 4.

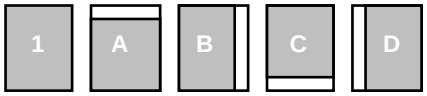


Figure 4 Subdividing a critical zone. Critical zone **1** is subdivided into the regions **A-D** by successively shrinking **1** by one pixel-width on each side.

Figure 5 contains an example space-scale diagram and corresponding screen-shots from ZTracker using the recursive critical zone algorithm.

Bottom-Up Computation

Critical zones can also be computed using a bottom-up algorithm. This pre-computes all possible critical zones for a given world. For all sets of (transitively) adjacent objects (including sets containing only one object), the scale at which the bounding box of the set is the size (along the largest dimension) of the window is computed and stored. This box will be a critical zone in all views that contain it and that are at a scale smaller than the computed scale. Figure 6 is a space-scale diagram indicating all possible critical zones of the size of the window. Note that some critical zones may reach window size at scales greater than the objects that they contain (indicated by horizontal dashed lines in the diagram). To maintain the integrity of critical zones—regions that contain interesting views—these are never indicated at scales larger than any of their contained objects.

Once the critical zones have been computed, a variety of methods can be used to select those that should be indicated in a view. Our single-zone algorithm yields the critical zone at the *smallest* scale that is contained in the current view. The recursive algorithm yields all critical zones in the current view that are *smaller* than the minimum zone-size, but at the *greatest* scale of any nested set of which they are a member.

VIEW-NAVIGATION ANALYSIS

The concepts used in ZTracker and the Landmarking system were inspired by an analysis of desert fog using *view-navigation theory* [3]. We present that analysis here in hope of providing a deeper understanding both of desert fog and of view-navigation theory. We apologize, in advance, for redundancies resulting from the earlier summary.

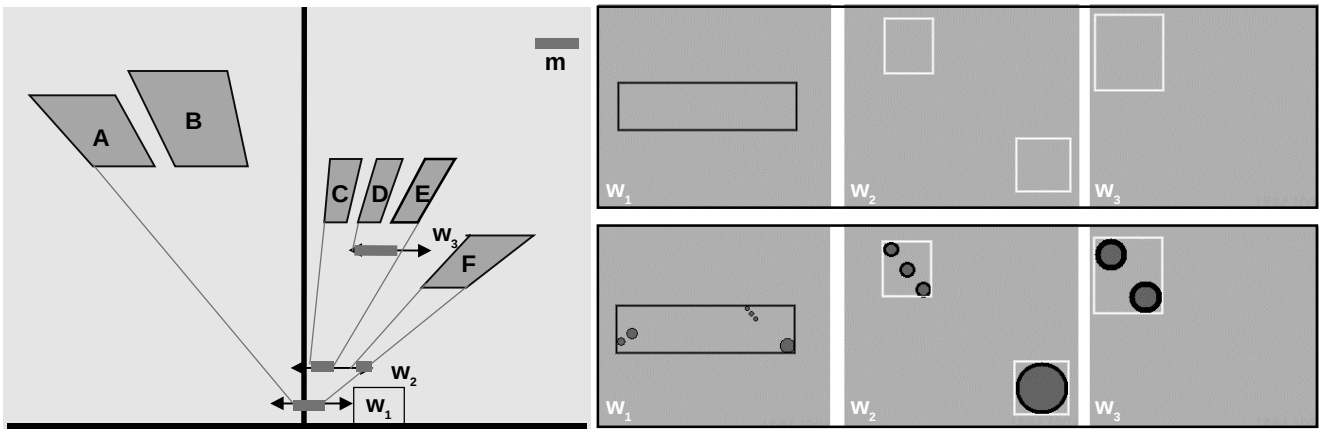


Figure 5 Multiscale residue in ZTracker using the Recursive Critical Zone algorithm. In the space-scale diagram, the heavily shaded portions of each view ($\mathbf{w_1}$, $\mathbf{w_2}$ and $\mathbf{w_3}$) denote the critical zones in that view. **m** indicates the minimum size for critical zones. (Note that **m** is artificially large for diagrammatic purposes. In actual use typical minimum sizes would be a small fraction of the window width.) In $\mathbf{w_1}$ the initial critical zone containing objects **A-F** is below the minimum size so is not refined further. On the screen, it is colored to indicate that $\mathbf{w_1}$ contains all objects in the world. In $\mathbf{w_2}$ the initial critical zone containing **C-F** is refined into the two critical zones containing **C-E** and **F**, respectively. In $\mathbf{w_3}$ the initial critical zone containing objects **D** and **E** is, again, below the minimum size so is not refined further. During zooming in, critical zones containing more than one object will expand until they are larger than the minimum size. They will then be replaced by multiple critical zones as has happened in $\mathbf{w_2}$.

View-navigation theory [3] provides a characterization of the properties that make an information structure navigable. It defines a view-navigable structure as a structure in which it is possible to determine how to get from any point to any other using only the information available in successive views. It was originally developed in the context of information worlds that are entirely defined by the information itself. We have had to extend the initial theory to accommodate spatial worlds in which the information is embedded in a pre-existing space. Our analysis demonstrates these extensions, but we do not attempt to formalize them here. It is important to note that this duality of embedded information and pre-existing space is one of the sources of desert fog in Pad++: Traversal—actual movement—is generally relative to the pre-existing space, while navigation—cognition about paths—is relative to the embedded information.

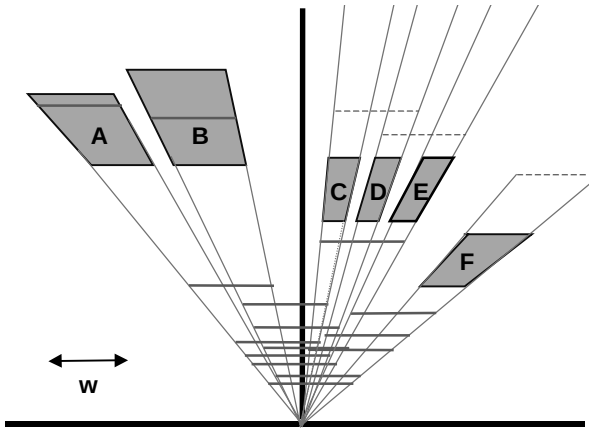


Figure 6 Bottom-Up computation of critical zones. Horizontal lines indicate window-sized critical zones. Dashed critical zones are those that are at a scale greater than any of their contained objects. There will be a critical zone for each set of (transitively) adjacent objects. A critical zone may appear in views from an infinitely small scale to the scale at which it becomes larger than the window.

Traversal Requirements

At the heart of view-navigation theory is a concept called the viewing graph. This is a directed graph in which nodes correspond to views in the information world, and links correspond to traversible connections between views. In a spatial multiscale world, views are portions of the pre-existing space—portions of the surface in Pad++—described by the location in space-scale of a given window. In a spatial multiscale world, links are single acts of traversal—in Pad++, a simple zoom or pan action. A link leading out of a view is an *out-going link* or *outlink*.

In order for a world to be traversible, two requirements must be satisfied:

1. Short paths must exist between all nodes, and
2. All views must have a small number of outlinks,

where "short" and "small" are relative to the overall viewing graph.

Furnas and Bederson [3] have shown that the first requirement is satisfied for spatial multiscale worlds. This is due to the ways in which a scale dimension affects the fundamental geometry of a space. For example, imagine adding a third spatial dimension to a two-dimensional space. The shortest distance between any two points is unaffected: it is still a straight line. Now imagine adding a scale dimension to the original two-dimensional space. The

shortest distance between two points may no longer be a straight line. Rather it may involve zooming out (moving through scale), panning slightly (moving in a spatial dimension), then zooming back in (moving through scale again). This "indirect" path through scale can be logarithmically shorter than moving at a fixed scale (pure panning). Moving through scale is analogous to navigating through a book by using the table of contents rather than flipping through the pages.

The second traversal requirement, on outlinks, holds in large information world that is usable through a (necessarily) resource-limited interface: Only a subset of all possible views in a spatial multiscale world is diagrammed in a space-scale diagram. In Figure 7, the views that can be reached, in a single step, from the view centered at **w** are those centered within the small gray region around **w**. This is a very small fraction of the diagrammed set of views, and an even smaller fraction of all possible views. There is exactly one outlink from a given view for each view reachable from that view, so each view clearly has a small number of outlinks relative to the number of views in the world.

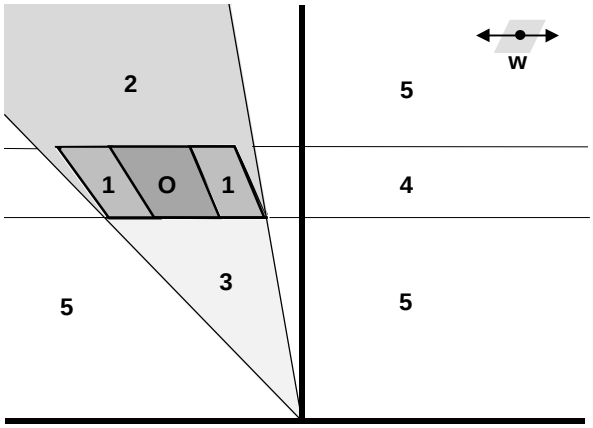


Figure 7 The anatomy of desert fog for pan/zoom interfaces such as Pad++. Regions **O** (the object itself) and **1** contain only non-desert fog views (also known as "interesting views"). To reach the object from region **2**, it is necessary to zoom out; from **3**, zoom in; from **4**, pan; from **5**, zoom and pan. (The gray area around **w**'s center shows the views reachable from **w**.)

Navigation Requirements

Navigational residue is evidence in a view that leads a navigator to believe that a particular target node may be reached by following a particular link. This composite concept consists of the clue provided by the environment along with the navigator's interpretation of and possible inferences based on that clue. *Good* residue is residue that correctly leads the navigator to believe that a shortest path to a node goes through a particular link, while *bad* (or *misleading*) residue does so erroneously. *Outlink-info* holds this navigational residue, producing clues to an outlink, e.g., in the form of a textual label.

In order for a world to be navigable, it must be traversible and satisfy two additional requirements:

3. All views must contain good residue of all nodes, and
4. All links must have small outlink-info.

In other words, any view must contain sufficient information to allow the navigator to choose correctly where to go next to get to[1] any desired node, and the amount of information about options presented in each view must be small. Note that "small" can now be interpreted relative to the number of overall views (as in the traversal requirements) or relative to the navigator's information-processing capabilities. We must, at least, satisfy the requirement relative to the overall number of views. We hope to satisfy it relative to the navigator's capabilities, at least in the average case.

[1] As originally defined, *strong* view-navigability [3] requires that "get to any node" is via a shortest path. In this paper, we relax this requirement and work with paths that can be shown to be within a small additive constant of shortest paths.

Spatial multiscale worlds do not fulfill either of these requirements. Desert fog views contain no outlink-info and, hence, no good residue. No outlink-info is small outlink-info, technically, but useless. We can thus re-cast the problem of addressing desert fog as a problem of providing good residue and small outlink-info in all views.

Providing Good Residue

Strictly speaking, there is a small amount of navigational information implicit in a desert fog view: The target object is elsewhere. This implicit outlink-info is called *improper* outlink-info to distinguish it from the normal, usually explicit *proper* outlink-info. Improper outlink-info, like proper outlink-info, is *good* outlink-info if it actually leads the navigator to the target.

In a spatial multiscale world, any outlink eventually leads to desert fog, so desert fog views collectively have good residue in all views. Since any particular desert fog view is made interesting only by its location relative to one or more interesting views, we focus on providing residue of interesting views. That is, we seek to provide good residue of all and only interesting views. We have two means of achieving this. We can provide good proper residue, or we can convert the bad improper outlink-info into good improper outlink-info. In other words, show the way to objects, or provide a single way to go if the way to a particular object is not indicated.

While a desert fog view tells the navigator that the target object is elsewhere, there is no way of knowing in which direction "elsewhere" lies. I.e., there is no way of knowing which type of desert fog the view contains. In Figure 7, we characterize different types of desert fog based on the actions required to get from a view to a given information object. This results in dividing the space into regions in which all views require the same action. (Views are defined

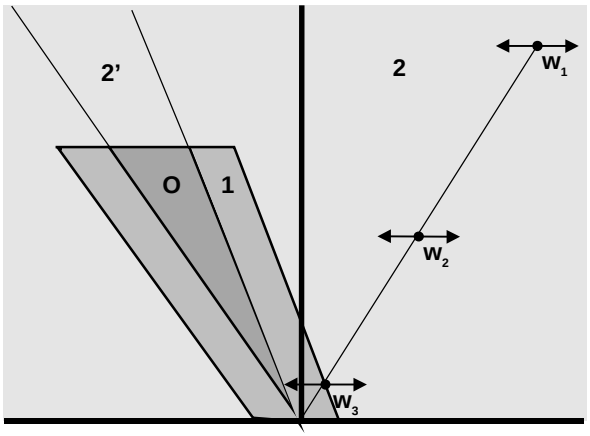


Figure 8 Anatomy of desert fog with infinite resolution viewer. A window zooming out moves along a line towards the origin and must eventually move from the desert fog region to region 1 where the object first appears at the edge of the window. Since this is the case for all views in the desert fog region, there is only one type of desert fog, type 2, with an unambiguous navigational action available. (In region 2', zooming out actually brings the window center over some part of the object.) The object has natural multiscale residue.

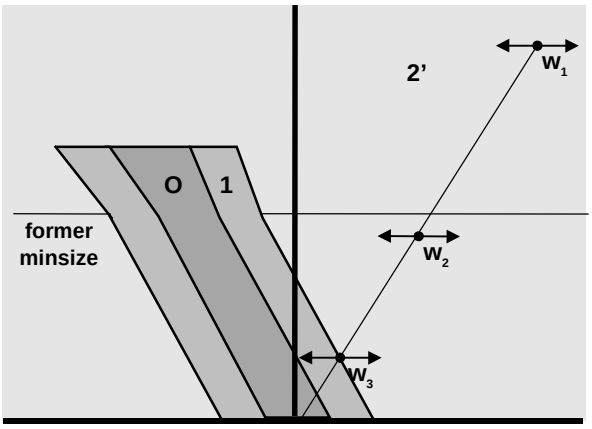


Figure 9 Object in a world with finite resolution, but no minimum size (e.g., objects are never allowed to disappear or get smaller than one or, as shown, more pixels). From infinite zoom-out to the scale of the object's former minsize the object is a constant size, then grows normally as the scale increases further. This is seen in the diagram as the parallel sides of the object begin to diverge at the former minsize scale. (In any view, zooming out brings the window center over some part of the object.) The object has artificial multiscale residue.

by their window centers.) The regions **O** and **1** do not contain desert fog views: within **O**, the window center is already somewhere over the object, and in region **1** (half a window wide) the object is still partially within the view. All other window positions are in regions of desert fog. They all look the same—no object is visible—but require different actions: If one is in region **2**, one would have to zoom out; in **3**, zoom in; in **4**, pan in some (unspecified) direction, in **5**, use some combination of zooming and panning.

However, if there were a single default way to go to find any object from desert fog ("when it looks like this go that-way"), the improper outlink-info would be sufficient and therefore good. If the viewing device could present, and the human viewer could perceive, infinite resolution, desert fog views would never contain objects. (All contained objects would be visible and the view would not be a desert fog view.) There would never be anything to zoom in on, so the correct navigational action in a desert fog view would always be to zoom out.

This infinite resolution scenario is diagrammed in Figure 8. If an object is in the view, it would have good proper residue. If it is not, it would have good improper residue. All objects would have multiscale residue (residue that reaches through scale). Multiscale residue can be emulated in a finite resolution world by not allowing objects to have a minsize. For example, we could always leave one or more pixels to represent the object, as illustrated in Figure 9.

Providing good residue by supplying multiscale residue for each object in a view solves the basic desert fog problem. If an object in the view is not visible for some reason, its multiscale residue will alert the navigator to its presence—stemming one source of desert fog. If there are no objects in the view, there will be no multiscale residue and the navigator will know to zoom out until something comes into view—alleviating the other source of desert fog. This approach for providing good residue depends on the converging property of zooming-out. Panning does not have a converging property, so the approach cannot be used in a pan-only world.

Providing Small Outlink-Info

Providing individual multiscale residue of every object is not a navigationally satisfying solution. Intuitively, views with large numbers of objects will be cluttered. Formally, view-navigation requirement 4 (small outlink-info) is violated. A single link potentially leads to all objects in the world so the outlink-info for that link would need, if every object is to have individual residue, to enumerate all objects. This outlink-info will be as large as the number of objects in the world. This may be small with respect to all possible views (meeting the traversal requirement), but may not be small with respect to the navigator's capabilities. The implication of view-navigation requirement 4 is that views must be grouped so that they can share outlink-info. We have explored two grouping methods in ZTracker and the Landmarking system.

FUTURE WORK

We are still seeking to understand the practical implications of view-based navigational aids, in general, and critical zones, in particular. We are planning user studies to test the sufficiency of critical zones for navigational purposes and to learn about the significance of their dynamic nature.

Along with basic usability, we would like to investigate the need for indicating nested critical zones and the need for refinement of overlapping critical zones. We are also planning on comparing the effect of aids using view-based grouping and aids using conventional clustering on users' perceptions of information worlds.

Our present work has concentrated on automatically showing *where* there are things of interest. The next major step of research is to show *what* those things are. Generation of appropriate labels is a difficult problem in semantically-based systems. Because of the dynamically changing groupings, we expect labeling to be even more difficult in view-based systems. Nonetheless, we recognize that we cannot achieve full navigability without labeling.

DISCUSSION

We have introduced the problem of desert fog and demonstrated a set of navigational aids that address its effects. Our aids are based on the concepts of multiscale residue and critical zone analysis. These concepts were derived from an analysis of desert fog using view-navigation theory in the context of spatial multiscale worlds. To accomplish this, we had to extend view-navigation theory to accommodate worlds that are defined by embedding information into a pre-existing space.

Critical zone analysis differs from conventional clustering techniques in three significant ways. First, because the resulting groups are view-dependent, groups change dynamically as the user moves through the world. Second, the resulting structure of groups is a semi-lattice rather than a hierarchical tree, meaning that overlapping groups are possible. Third, and most importantly, the analysis does not assume that the spatial layout of the information is meaningful, nor does the resulting grouping convey an impression of semantic coherence within groups.

Desert fog, as presented here, is representative of a larger class of *desert-fog-like* problems wherein a view contains navigational clues, but these are inaccessible to the navigator. For instance, they may be obscured or lost in visual clutter. Combining critical zone analysis with filtering mechanisms may serve to alleviate such problems.

Desert fog and desert-fog-like problems abound in spatial multiscale worlds regardless of their interaction metaphor. For instance, being blocked—and not knowing why—by an object that fills the view is a known problem in 3D walk-through worlds. In such worlds, it may be desirable to use view-based navigational aids to indicate where things are *not*, e.g., "critical zones" that show directions in which it is *possible* to move. Whether view-based navigational aids can be applied to other types of worlds remains to be seen.

We hope that we have provided a general way of thinking of desert fog problems that can lead to other types of solutions. We also hope that we have provided a useful example of solving a navigational problem using view-based navigational aids.

ACKNOWLEDGEMENTS

This work was supported, in part, by ARPA Grant N66001-94-6039, and by an Irma M. Wyman grant from the Center for the Education of Women at the University of Michigan. We offer our special appreciation to Rudy Darken and Dan Russell for their comments on earlier drafts.

REFERENCES

1. Bederson, B. B., Hollan, J. D. (1994). Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. *Proceedings of ACM UIST'94*, New York, NY: ACM Press, 17-26.
2. Everitt, B. (1980). *Cluster Analysis.* 2nd Ed. New York: Halsted Press.
3. Furnas, G. W. (1997). Effective View-Navigation. *Human Factors in Computing Systems CHI '97 Conference Proceedings*, New York, NY: ACM Press, 367-374.
4. Furnas, G. W., Bederson, B. B. (1995). Space-Scale Diagrams: Understanding Multiscale Interfaces. *Human Factors in Computing Systems CHI '95 Conference Proceedings*, vol. 1, New York, NY: ACM Press, 234-241.
5. Henderson, D. A., Card, S. K. (1986). Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface. *ACM Transactions on Graphics*, 5 (3, Jul.), 211-243.
6. Ingram, R., Benford, S. (1995). Legibility Enhancement for Information Visualisation. *Proceedings of IEEE Visualization '95*. 209-216.
7. Jul, S., Furnas, G. W. (1997). Navigation in Electronic Worlds. *SIGCHI Bulletin*, 29, 4 (Oct), 44-49.
8. Hirtle, S. C., Jonides, J. (1985). Evidence of Hierarchies in Cognitive Maps. *Memory & Cognition*, 13(3), 208-271.
9. Hirtle, S. C., Mascolo, M. F. (1986). Effect of Semantic Clustering on the Memory of Spatial Locations. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 12, 182-189.
10. Kullberg, R. L. (1995). Dynamic Timelines: Visualizing the History of Photography. *Human Factors in Computing Systems Conference Companion CHI 96*, New York, NY: ACM Press, 386-387.
11. Masui, T., Minakuchi, M., Borden, G. R. IV, Kashiwagi, K. (1995). Multiple-View Approach For Smooth Information Retrieval. *Proceedings of the ACM Symposium on User Interface Software and Technology '95*, New York, NY: ACM Press, 199-206.
12. Sadalla, E. K., Burroughs, W. J., Staplin, L. J. (1980). Reference Points in Spatial Cognition. *Journal of Experimental Psychology*, 6, 516-528.