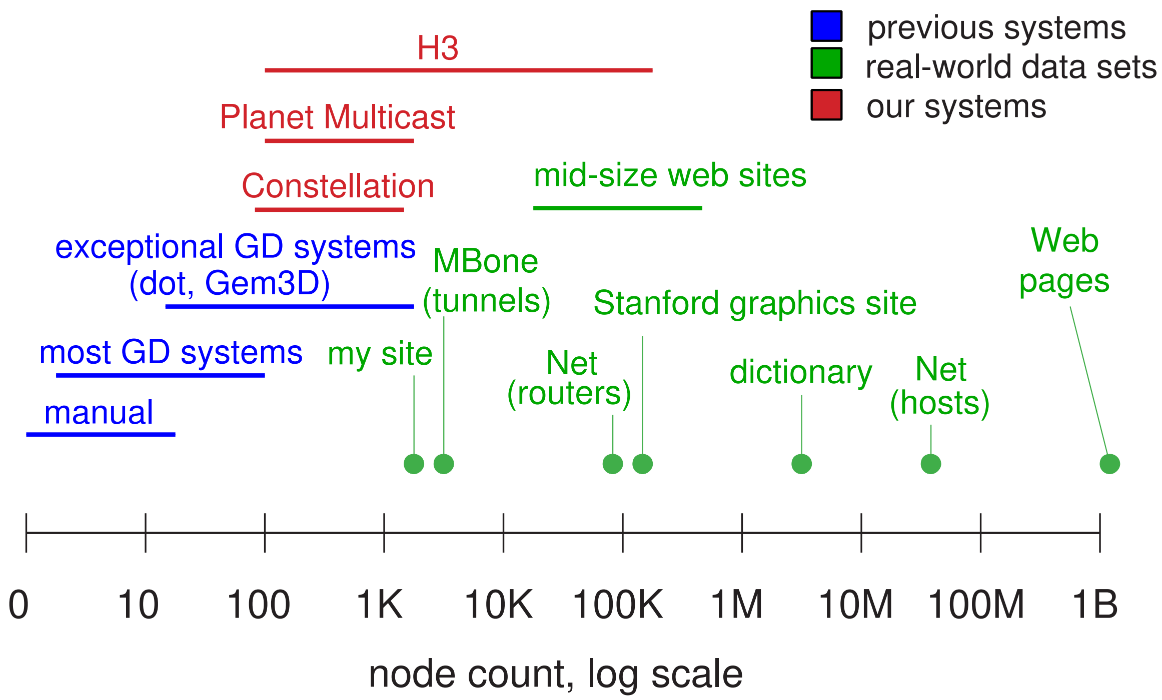

Figure 1.2: **System scalability and dataset size.** Previous graph drawing systems, shown in blue, fall far short of many large real-world datasets, shown in green. The three systems in this thesis, shown in red, start to close this gap by aiming at datasets ranging from thousands to over one hundred thousand nodes.

evidence. In the Constellation chapter we discuss the influence of our informal usability observations on the system design, in addition to a heavy emphasis on the conceptual framework analysis.

1.3.3 Scalability

Very small graphs can be laid out and drawn by hand, but automatic layout and drawing by a computer program can scale to much larger graphs, and provides the possibility of fluid interaction with resulting drawings. The goal of these automatic graph layout systems is to help humans understand the graph structure, as opposed to some other context such as VLSI layout. Researchers have begun to codify aesthetic criteria of helpful drawings, such as minimizing edge crossings and emphasizing symmetry [BMK95, BT98, DC98, PCJ95, Pur97].

However, almost all previous automatic graph drawing systems have been limited to small datasets. The scalability discrepancy between systems for nonvisual graph manipulation and those designed to create visual representations of them is attributable to the difficulty of general graph layout. Most useful operations for drawing general graphs have been proved to be NP-complete [Bra88]. Most previous systems are designed to

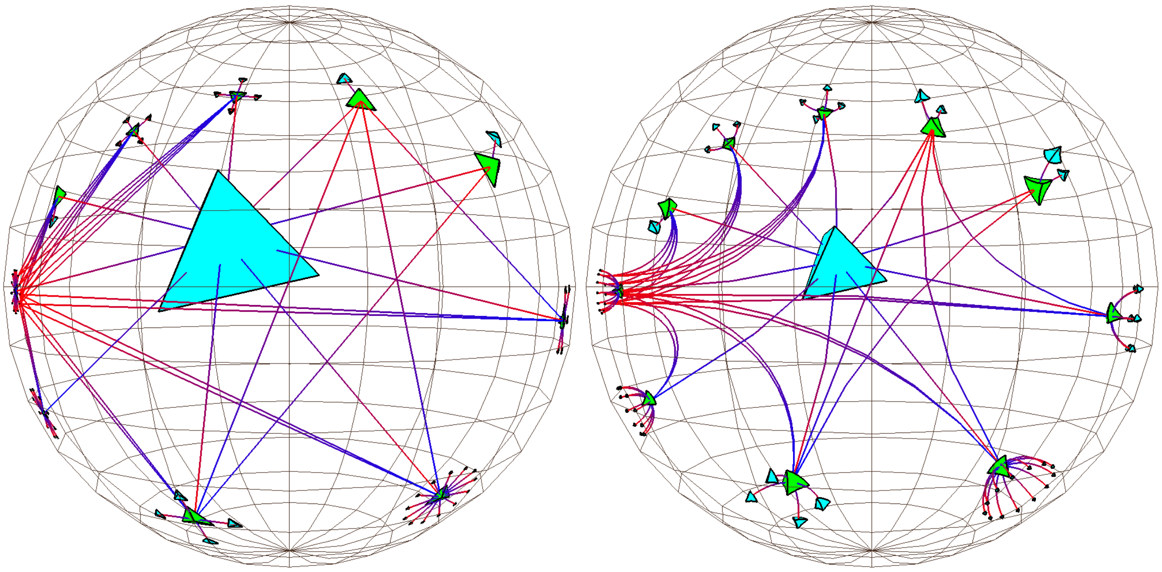

Figure 3.5: **Models of hyperbolic space. Left:** The projective model of hyperbolic space, which keeps lines straight but distorts angles. **Right:** The conformal model of hyperbolic space, which preserves angles but maps straight lines to circular arcs. These images were created with the *webviz* system from the Geometry Center [MB95], a first attempt to extend cone tree layouts to 3D hyperbolic space that had low information density. The cone angle has been widened to 180°, resulting in flat discs that are obvious in the projective view. The arcs visible in conformal view are actually distorted straight lines.

can use these standard libraries for efficient computation of our hyperbolic transformations in the projective model, since they can conveniently also be encoded as 4×4 matrices. In contrast, transformations in the conformal model require 2×2 complex matrices, which cannot be trivially transformed into a more standard operation. Moreover, standard graphics libraries are much slower when drawing curved objects than flat ones, since curves are approximated by many piecewise linear lines or polygons. Since the main goal of the H3 system was scalability to large datasets, we chose to use the projective model. An extensive discussion of 3D hyperbolic projective transformations for use in computer graphics appears in a paper by Phillips and Gunn [PG92], so we do not discuss these in detail here.

The procedure for projecting from hyperbolic to Euclidean space under the Klein-Beltrami model is easiest to understand by first looking at the one-dimensional case shown in Figure 3.6. The entire hyperbola can be projected onto a finite Euclidean open line segment tangent to its pole, using an eye point one unit below that pole. The asymptotes of the hyperbola are the boundaries of the line segment and cross at the eye point. Objects (in this case, 1D lines) drawn on the hyperbola itself can be projected to the line segment. The projections of objects exactly at the pole are undistorted, but objects far from the pole project to a vanishingly small part of the line segment. Rigid hyperbolic objects that translate along the hyperbola will appear in the

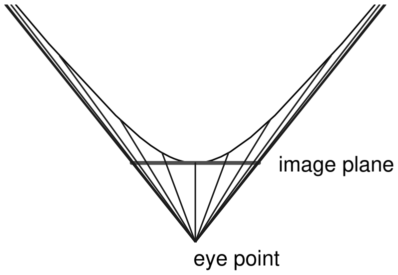

Figure 3.6: **1D hyperbolic projection.** An illustration of the projective model for one-dimensional hyperbolic space. The image plane is at the pole of one sheet of the hyperbola and the eye point is where the asymptotes meet. Although the projection near the pole is almost undistorted, the apparent shrinkage increases as the rays reach further up the hyperbola.

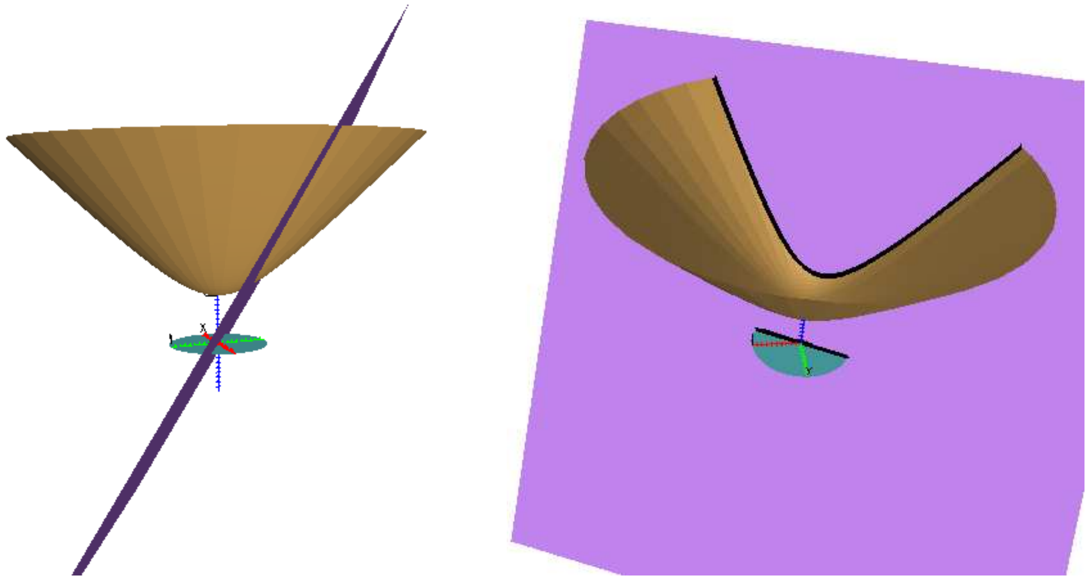

Figure 3.7: **2D hyperbolic projection.** An illustration of the projection from a unit hyperboloid to a Euclidean disc at $z = 0$. **Left:** A cutting plane which goes through the origin is shown nearly edge-on: it intersects the hyperboloid in a geodesic, and the disc in a straight line. **Right:** We show the same scene from a different angle, so that the purple cutting plane is nearly parallel to the image plane.

projection to grow to a maximum when at the pole and then shrink, and after translating infinitely up the hyperbola their projection will be infinitely close to the line segment border.

Figure 3.7 shows the two-dimensional case: the surface of the hyperboloid projects into a disc – a finite section of a Euclidean plane. The plane is arranged so that it cuts the hyperboloid in a geodesic line, which

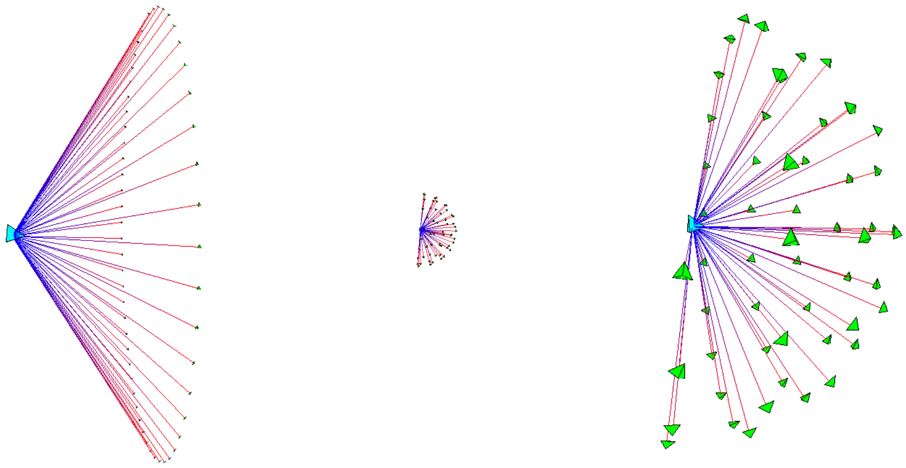

Figure 3.8: **Circumference vs. hemisphere.** The H3 layout on the surface of a spherical cap at the mouth of a cone is much more compact than the traditional cone tree layout on the circumference of the cone mouth. All three pictures show 54 child nodes in hyperbolic space. **Left:** The traditional perimeter layout requires a large cone radius, such that it is quite sparse. **Middle:** The hemispherical layout is scaled so that the child node sizes and cone radius can be directly compared with the perimeter image on the left. **Right:** The hemispherical layout is scaled so that the parent node size can be directly compared with the perimeter image on the left.

However, we can draw a tree in hyperbolic space compactly while allocating the same amount of hyperbolic space for each leaf node no matter how deep it is in the tree. The root node is then the same size as any other node and is special only in that it has no parent. The entire tree structure has roughly equal local information density. Setting the focus for the visualization is simply a matter of translation on the 3-hyperboloid to bring the desired node to the pole. Its projection at the origin of the ball will be large, and many of the nearby nodes will be visible for additional context.

The H3 algorithm can be thought of as an extension of the influential Cone Tree system developed at Xerox PARC [RMC91]. The Cone Tree algorithm lays out tree nodes hierarchically in 3D Euclidean space, with child nodes placed on the circumference of a cone emanating from the parent. The *webviz* project at the Geometry Center [MB95] was a first attempt to adapt the Cone Tree algorithm for use in 3D hyperbolic space. The *webviz* layout, shown in Figure 3.5, used a cone angle of $180°$, widening the cone into a flat disc. That algorithm did exploit some of the exotic properties of 3D hyperbolic space, but did not unleash its full potential: the amount of displayed information compared to the amount of white space was quite sparse.

Our H3 algorithm instead lays out child nodes on the surface of a hemisphere that covers the mouth of a cone, like sprinkles on an ice cream cone. Figure 3.8 compares this layout to the original cone tree approach. Like the *webviz* algorithm, we use a cone angle of $180°$ to subtend an entire hemisphere. The

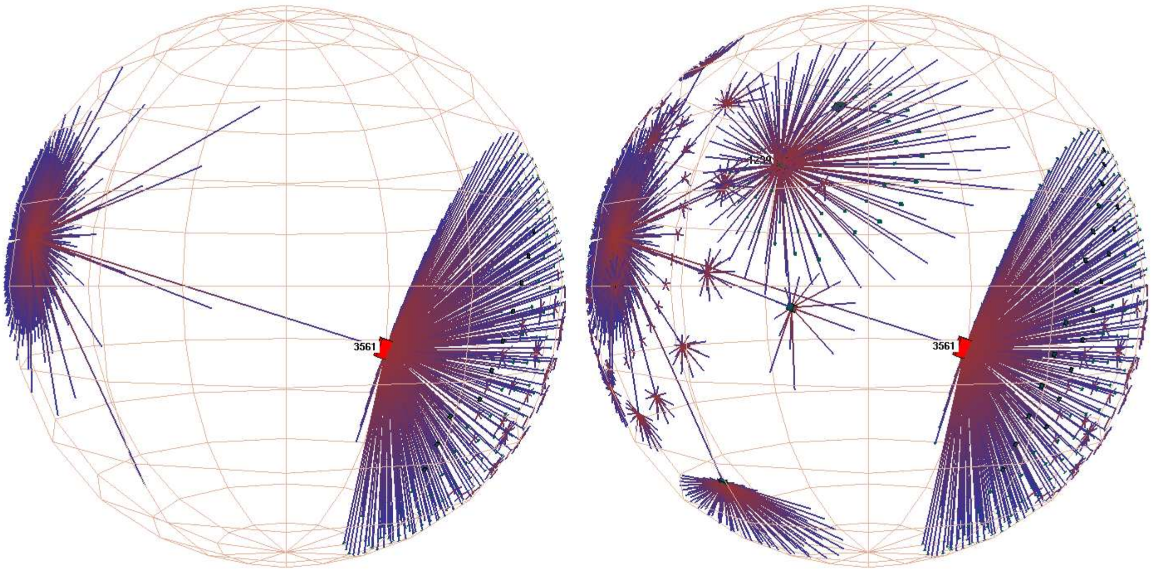

Figure 3.13: **Active vs. idle frames, obvious case.** The H3Viewer guaranteed frame rate mechanism ensures interactive response for large graphs, even on slow machines. **Left:** A frame drawn in 1/20th of a second during user interaction. **Right:** A frame filled in by the idle callbacks for a total of 2 seconds after user activity stopped. The graph shows the peering relationships between Autonomous Systems, which constitute the backbone of the Internet.[8] The 3000 routers shown here are connected by over 10,000 edges in the full graph.

Although the hop count between two nodes does not change during navigation, the projected screen area of a node depends on the current position of the graph in hyperbolic space. Navigation occurs by moving the object in hyperbolic space, which is always projected into the same fixed area of Euclidean space. Motion on the surface of the 3-hyperboloid brings some node of the graph closest to the pole, such that its projection into the ball is both closest to the origin and largest. Most previous systems for adaptive drawing [Hop97] state the viewpoint problem in terms of camera location, but for convenience in our case we consider instead the mathematically equivalent problem of an object position with respect to a fixed camera.

3.3.1.2 Active, Idle, and Pick Modes

When the user is actively interacting with a scene, frames much be drawn quickly in order to provide a feeling of fluid responsiveness. However, many scenes contain more data than can be drawn in a single brief frame. Our adaptive drawing algorithm tunes the amount of work which is attempted during a frame depending on the activity of the user, and takes advantage of idle periods of time between spurts of user interaction to fill

[8] The Autonomous System data was provided by David M. Meyer of the University of Oregon Route Views Project. `http://antc.uoregon.edu/route-views`

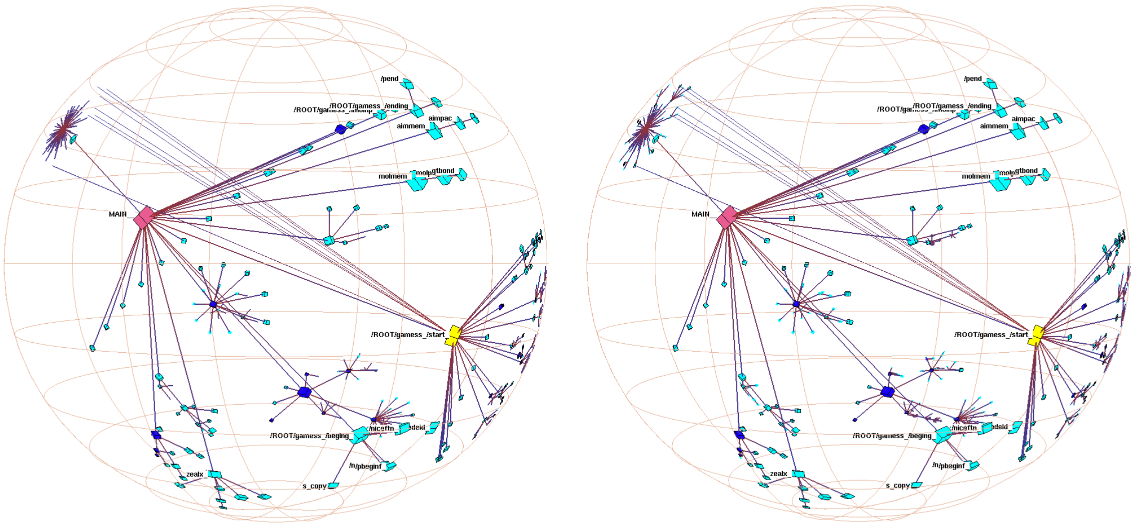

Figure 3.14: **Active vs. idle frames, subtle case.** A function call graph of a small FORTRAN benchmark with 1000 nodes and 4000 edges. Non-tree links from one of the functions are drawn. **Left:** a single frame has been drawn in 1/20th of a second. Note that the non-tree links to the distant fringe are visible even though their terminating nodes were small enough that the active drawing loop terminated before they could be drawn. Our drawing algorithm thus hints at the presence of potentially interesting places that the user might wish to drag toward the center to see in more detail. **Right:** The entire graph is drawn after the user has stopped moving the mouse. The graph is small enough that the difference between the two frames is more subtle than in Figure 3.13, and is visible only in the fringe in the upper left corner.

in more of the scene detail.

The H3Viewer library allows separate control over the drawing frame time, picking frame time, and idle frame time. The rendering frame time is simply the time budget in which to draw a single frame during user mouse movement or an animated transition. To ensure fluid interaction, the drawing time should be explicitly bounded instead of increasing as the node-link count rises. The time spent casting pick rays into the scene must be similarly bounded. Rendering and picking should execute somewhere between five and thirty frames per second (FPS) – our current default is 20 FPS for rendering and 10 FPS for picking. When the user and application are idle, the system can fill in more of the surrounding scene. On a high-end system, or with graphs of only moderate size, the distant fringe is filled in unobtrusively, as in the top row of Figure 3.13. On a low-end platform, or with larger graphs, the difference between active and idle modes is more noticeable, as in Figure 3.14.

The time spent filling in the fringe when the user is idle can also be explicitly bounded. This limit is relevant only if there is some urgent need to immediately free the CPU for other tasks, for instance, if a high resolution H3 window is linked with several other interactive views under an umbrella application, and one

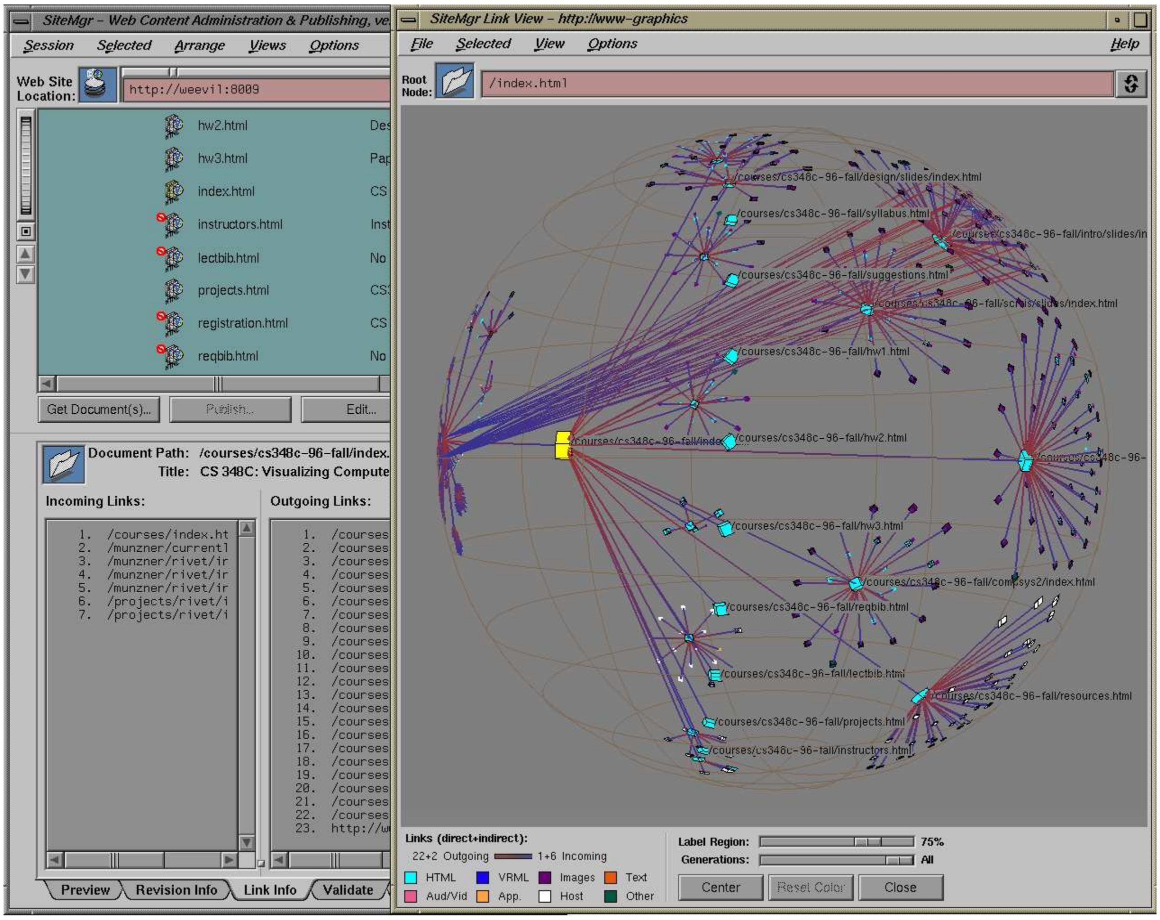

Figure 3.15: **Site Manager.** The Site Manager product for webmasters from SGI incorporated the H3 and H3Viewer libraries. The 3D hyperbolic view of the hyperlink structure of the web site is tightly linked with a traditional 2D browser view of its directory structure. Figure 3.20 shows the same dataset in a standalone viewer.

the outline view or selecting an entire subtree in the graph view, they are highlighted in both but no motion occurs. The support for brushing allows users to correlate information across views, as shown by the cyan circles in Figure 3.16. The figure also demonstrates how linked views can trigger functionality in other software components through direct manipulation of a graph view, since clicking on the parent node in the subtree resulted in an HTML preview of the associated document in the lower left corner.

The H3Viewer is optimized for browsing, but the tradeoff is that it is quite frustrating to search by navigation for a node when the user knows the name but its spatial location. Applications built for tasks where searching by name is desirable should link the H3Viewer window with a user interface element suitable for

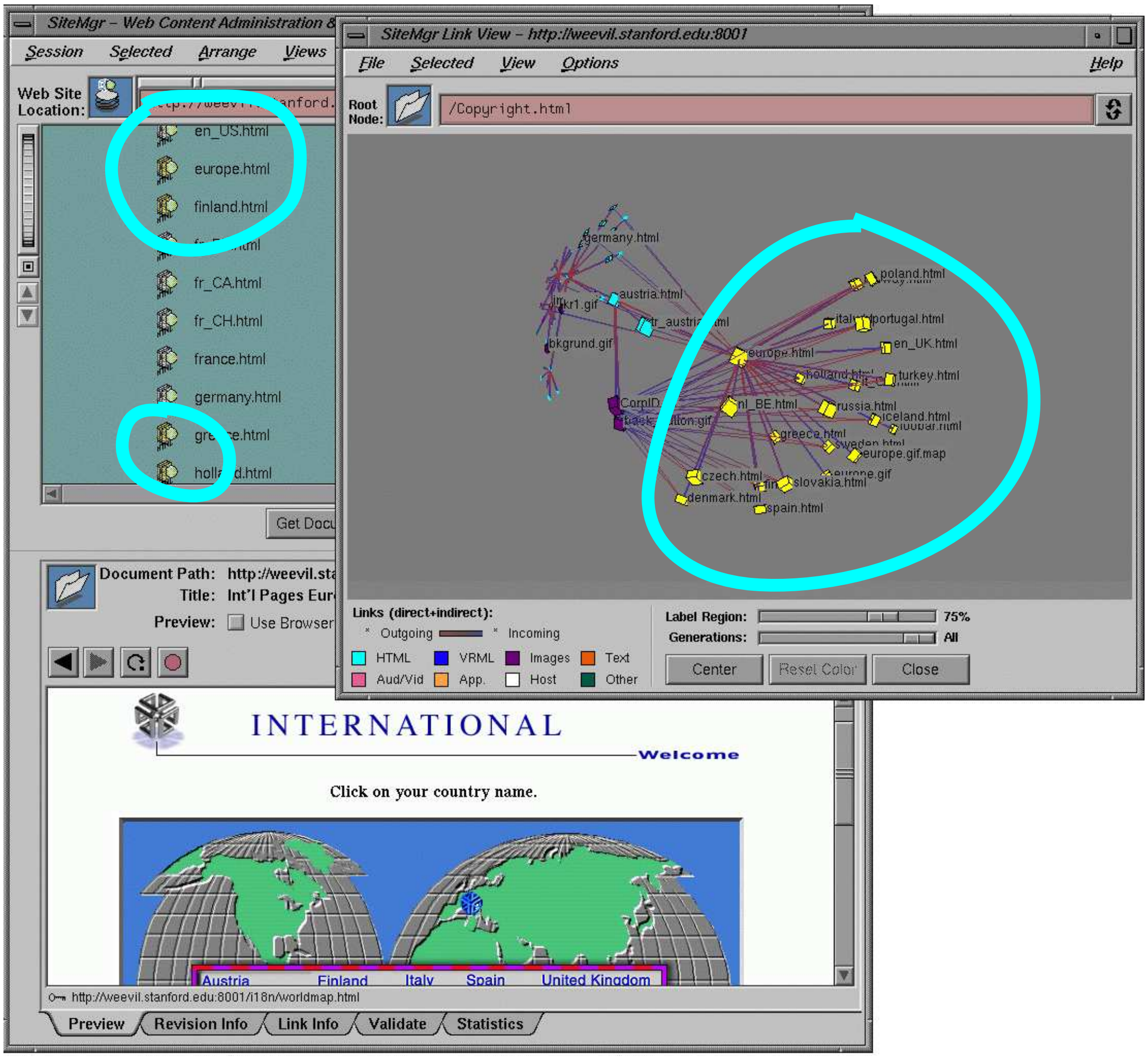

Figure 3.16: **Linked views.** The H3 library allows brushing. The highlighted subtree in the H3 view on the right shows the spatial proximity of a set of documents laid out according to their hyperlink structure, whereas the visible nodes in the directory view on the left that were highlighted because of the link are not all spatially proximate. Also, the HTML preview in the lower left corner was triggered by direct manipulation of the subtree parent node in the H3 view.

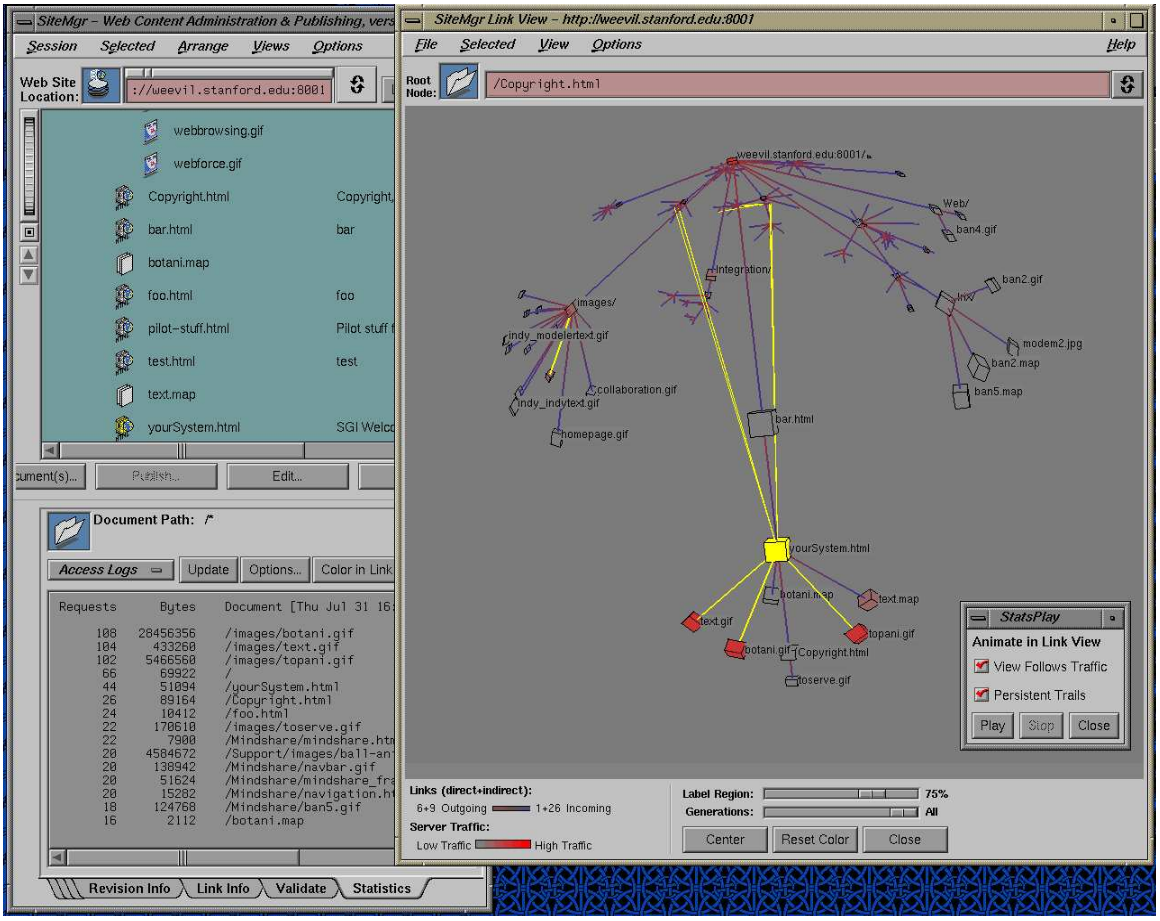

Figure 3.17: **Traffic log playback.** The Site Manager application controls the linked H3Viewer window through the API, so that each hit from the site's traffic log triggers an animated transition and a series of color and linewidth changes. The result looks like the source document fires a laser beam at the destination, which ends up slightly hotter afterward. The nodes are all colored grey at the beginning of the playback, and by the end the most popular documents are a fully saturated red.

handling search, and use the H3Viewer library API to trigger highlighting and animated transitions as appropriate. The Site Manager application has a search window with an input field for typing a query term and a display for the URLs returned by the query. Clicking on a result in that display triggers selection and animated transition in the H3Viewer. The user can then use the navigational capabilities of the H3Viewer to investigate the local area.

Another example of the power of linked views is shown in Figure 3.17. The Site Manager application can analyze a web site's traffic logs. The analysis window on the lower right contains a flat list of the most popular pages. The H3Viewer window on the right is being controlled through the API to show a hit by

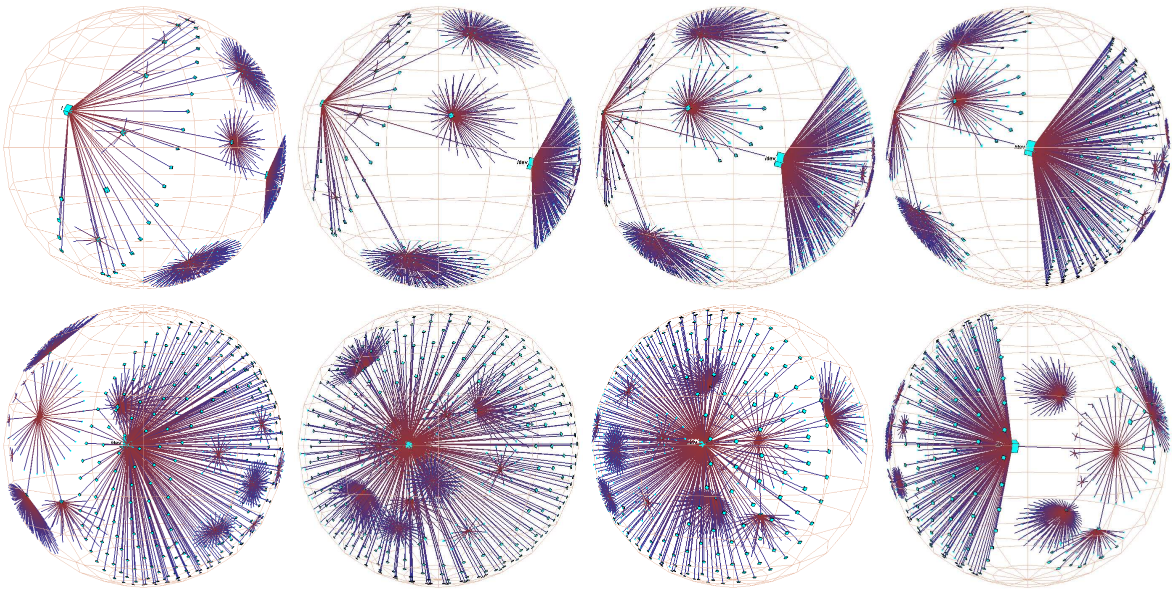

Figure 3.18: **Hyperbolic motion over a 30,000 element Unix file system.** Many nodes and edges project to subpixel areas and are not visible. **Top row:** Translation of a node to the center. **Bottom row:** Rotation around the center node. The rotation clarifies that objects lie inside a ball, not on the surface of a hemisphere. The file system has a strikingly large branching factor when compared with the web sites in Figure 3.20 or the call graphs in Figure 3.21. The directory that approaches the center, `/usr/lib`, contains a large number of files and subdirectories.

an animated transition is guaranteed to be constant, just like the frame rate during explicit user navigation. The transformation includes both a translation to move the node from its old position to the origin, and a rotational component to position the node so that its ancestors are all on the left side and its descendants are all on the right. This configuration provides a canonical local orientation. We add a slight tilt so that the ancestor-descendant axis is not perfectly horizontally aligned with the principal axis of the window, so that the text labels are less likely to occlude each other.

3.6.1 Navigation

If the user clicks on an edge, that point is translated to the center of the sphere but no rotation or selection occurs. The user can also rotate the structure around the origin of the ball, which spins everything around the current focus node. Finally, power users can have explicit control over hyperbolic translation through mouse drags. Figure 3.18 shows a multi-image sequence of hyperbolic translation.

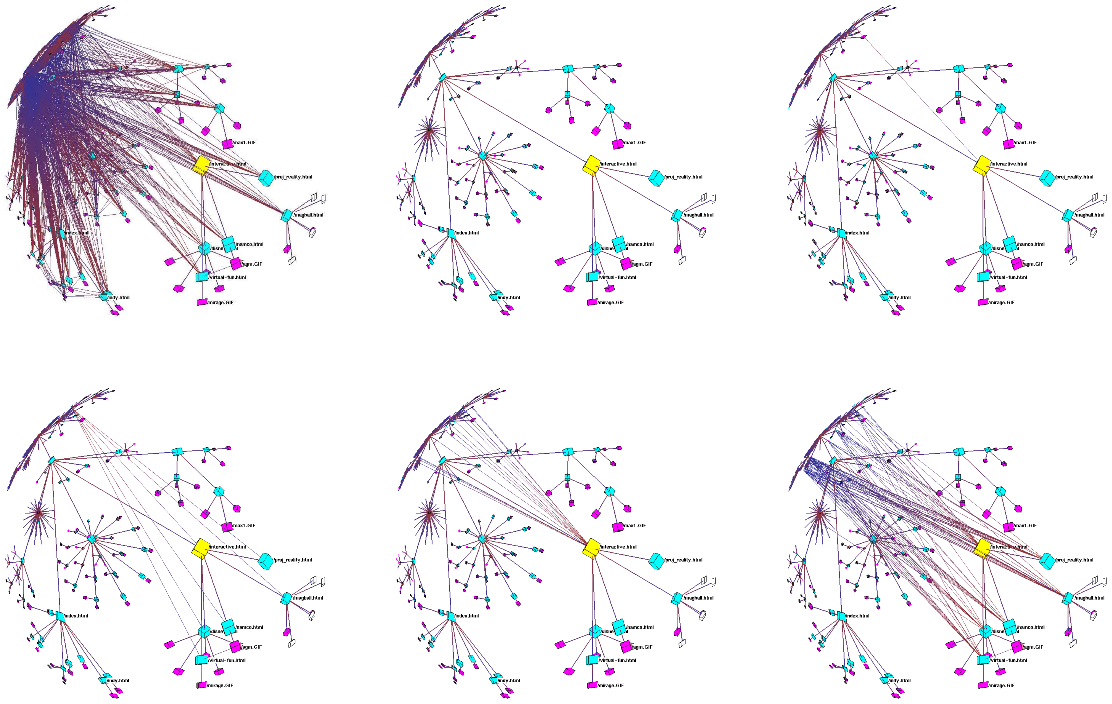

Figure 3.19: **Non-tree links. Top left:** Drawing all the non-tree links for dense quasi-hierarchical graphs results in an incomprehensible mess. **Top middle:** The spanning tree alone is quite clear. **Top right:** The node highlighted in yellow near the center has a single incoming non-tree link. **Bottom left:** The entire subtree beneath that node has more incoming non-tree links. **Bottom middle:** The highlighted yellow node has many outgoing non-tree links. **Bottom right:** Drawing the outgoing non-tree links for the entire subtree beneath the highlighted yellow node shows the highly interconnected nature of the web site without overwhelming the viewer with a mass of links.

3.6.2 Non-tree links

Non-tree links, which are in the original graph but not in the computed spanning tree, do not affect the layout. These non-tree links are drawn by simply connecting the two nodes that have been laid out in the tree structure. We do not explicitly check for edge crossings because our tree layout makes such a crossing in 3D space highly unlikely.

Although the spanning tree links are always drawn, the non-tree links are drawn only on demand. The top left image of Figure 3.19 shows why: a typical web site with all the non-tree links drawn results in an incomprehensible mess. The ability to interactively select which non-tree links are drawn is critical for making the H3 layout useful with dense quasi-hierarchical graphs. The user can draw incoming or outgoing non-tree links on demand for any node or subtrees. The rest of Figure 3.19 compares the visual effect of

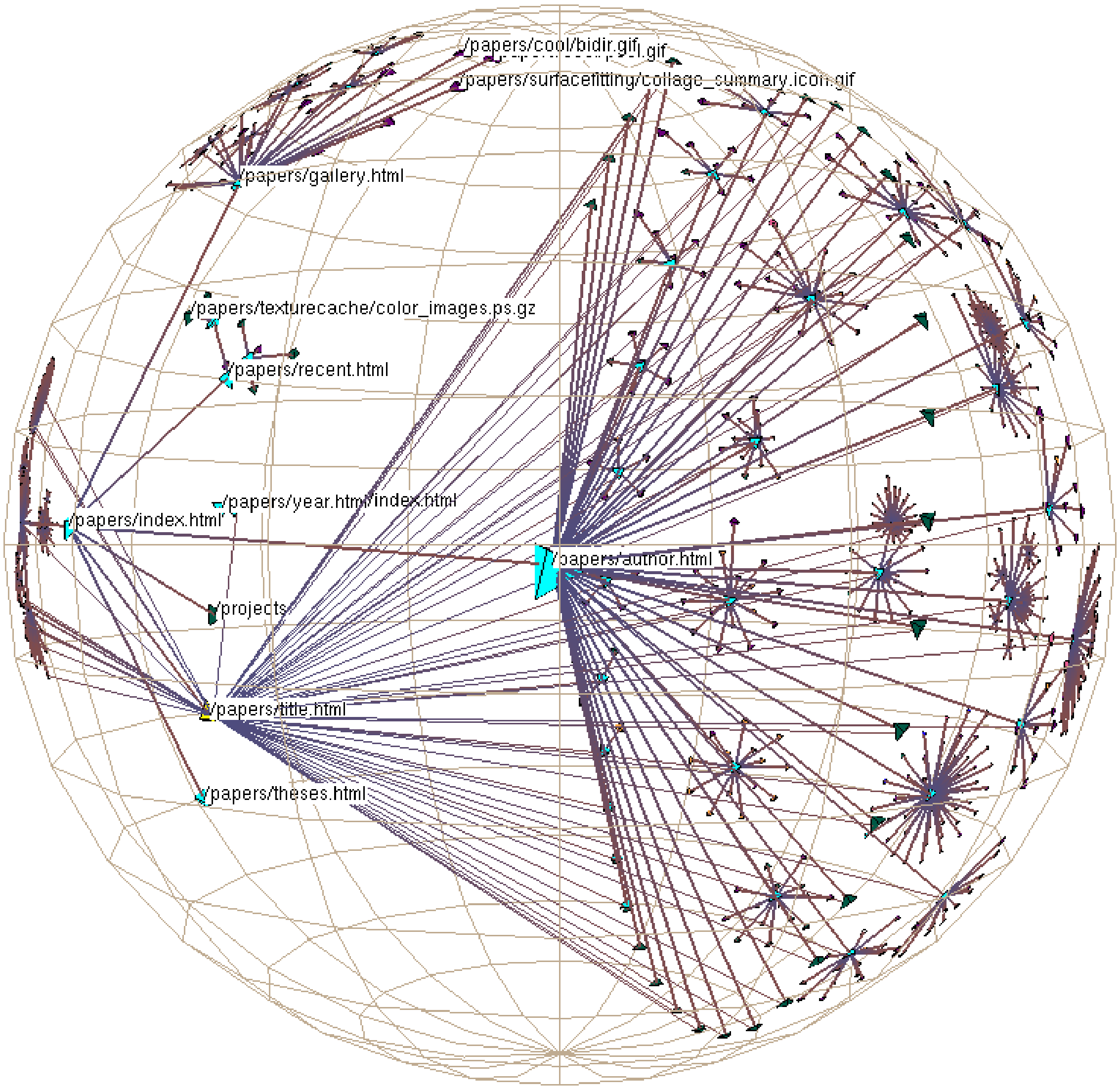

Figure 3.20: **Part of the Stanford graphics group web site drawn as a graph in 3D hyperbolic space.** The entire site has over 14,000 nodes and 29,000 edges. About 4000 nodes in a 7-hop diameter neighborhood of the papers archive are visible in this snapshot. One dozen of the nodes are labelled, a few hundred more have visible color coding or are individually distinguishable, and the rest of the nodes on the fringe provide aggregate information about their presence or absence. In addition to the main spanning tree, we draw the nontree outgoing links from an index of every paper by title. The tree is oriented so that ancestors of a node appear on the left and its descendants grow to the right.

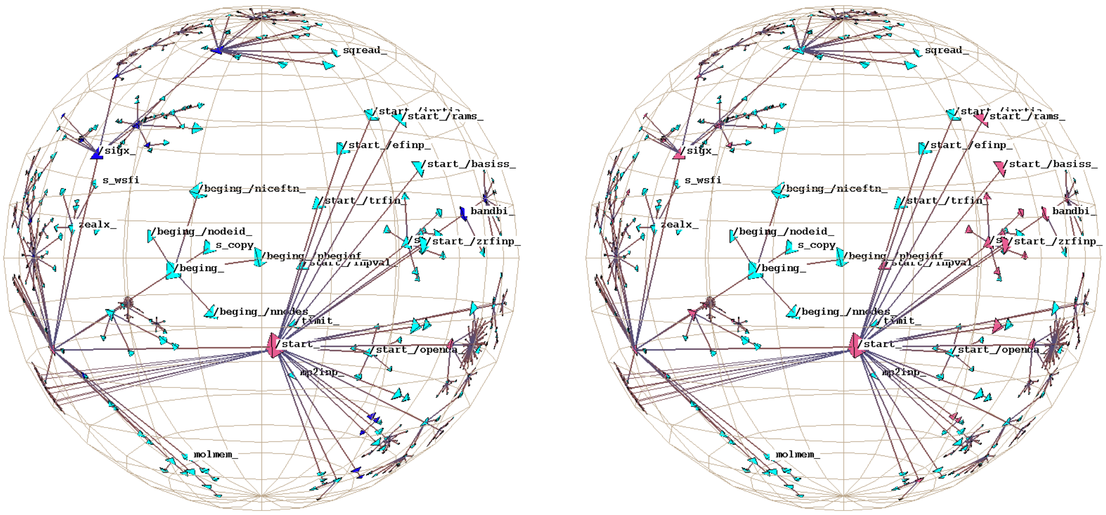

Figure 3.21: **Call graph.** We show the static function call graph structure for a mixed C and FORTRAN scientific computing benchmark. On the left the node coloring indicates whether a particular global variable was untouched (cyan), referenced (blue), or modified (pink). On the right we use the same color scheme with a different variable. Such displays can help software engineers see which parts of a large or unfamiliar program might be modularizable.

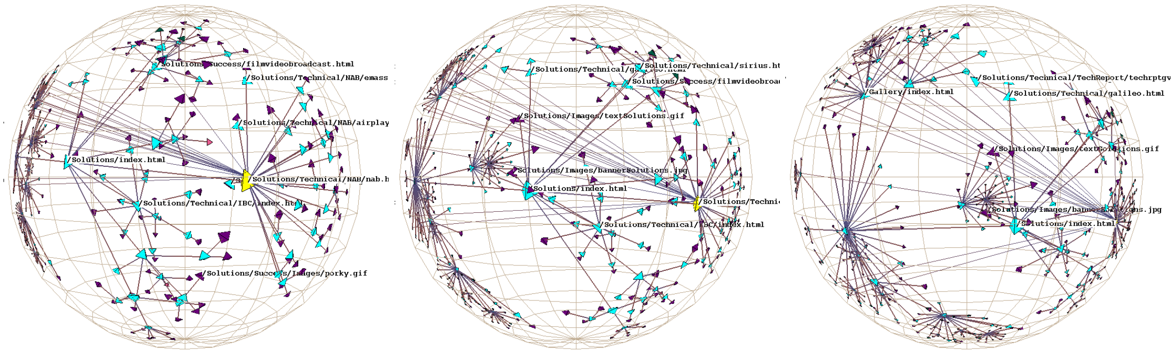

Figure 3.22: **Link structure of a web site laid out in 3D hyperbolic space.** We show the link structure of a web site laid out in 3D hyperbolic space. The nodes represent documents, which are colored according to MIME type: HTML is cyan, images are purple, and so on. We draw the outgoing non-tree links for the selected node, highlighted in yellow. **Top:** The destination of the outgoing links is quite distorted, but we do see that most of the links end at a particular cluster. **Bottom left:** We drag that cluster towards the focus to see more detail. **Bottom right:** The cluster is close enough to the center to see individual destination nodes.

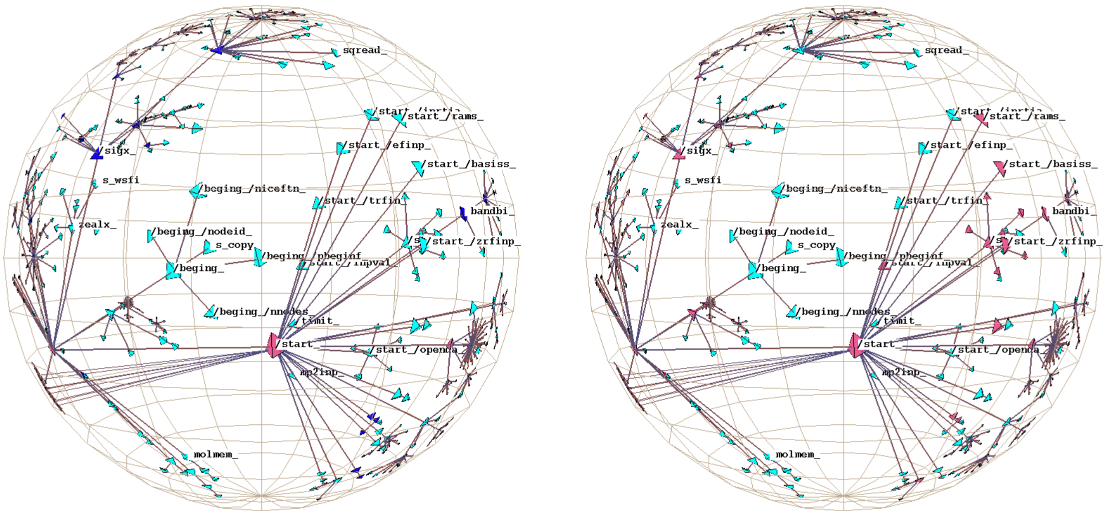

Figure 3.21: **Call graph.** We show the static function call graph structure for a mixed C and FORTRAN scientific computing benchmark. On the left the node coloring indicates whether a particular global variable was untouched (cyan), referenced (blue), or modified (pink). On the right we use the same color scheme with a different variable. Such displays can help software engineers see which parts of a large or unfamiliar program might be modularizable.

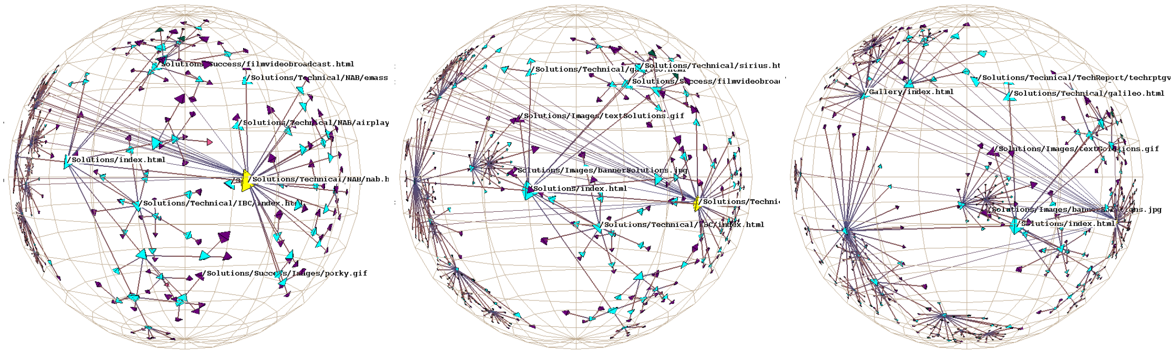

Figure 3.22: **Link structure of a web site laid out in 3D hyperbolic space.** We show the link structure of a web site laid out in 3D hyperbolic space. The nodes represent documents, which are colored according to MIME type: HTML is cyan, images are purple, and so on. We draw the outgoing non-tree links for the selected node, highlighted in yellow. **Top:** The destination of the outgoing links is quite distorted, but we do see that most of the links end at a particular cluster. **Bottom left:** We drag that cluster towards the focus to see more detail. **Bottom right:** The cluster is close enough to the center to see individual destination nodes.

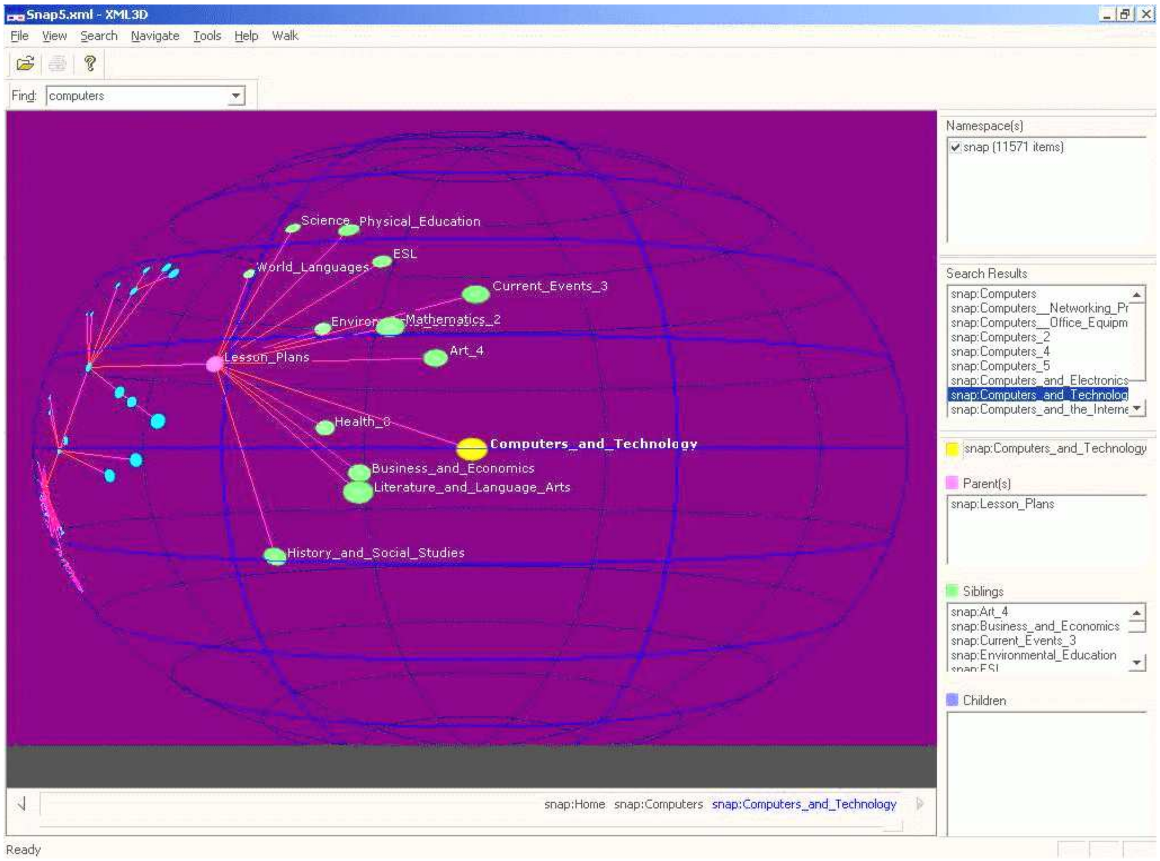

Figure 3.23: **XML3D interface.** The XML3D browser, which combined an H3Viewer window with several auxiliary lists, was compared with two traditional web browsers during a user study. XML3D was reliably faster for some tasks and had equivalent performance for others. Image courtesy of Mary Czerwinski, Microsoft Research.

XML3D also includes a history list and search capability. In Section 3.5.1, we asserted that the H3Viewer is most powerful when effectively linked with other interface components. XML3D was built to explicitly test this assertion in a particular task domain, that of Web content developers and maintainers, our target audience. Figure 3.23 shows an example of the XML3D application.

We had access to the 12,000 node categorization hierarchy which was available from the `snap.com` portal, and ported its contents into XML3D and a 2D collapsible tree browser. We used the live Snap.com hierarchy as the second 2D alternative during the study. Tasks with different levels of complexity were created by varying: (1) whether the subject was asked to find an existing category or add a new category to the directory scheme; and (2) whether the target category and requested response involved a single parent path or multiple parent paths. The tasks varied in kind as well as complexity across levels.

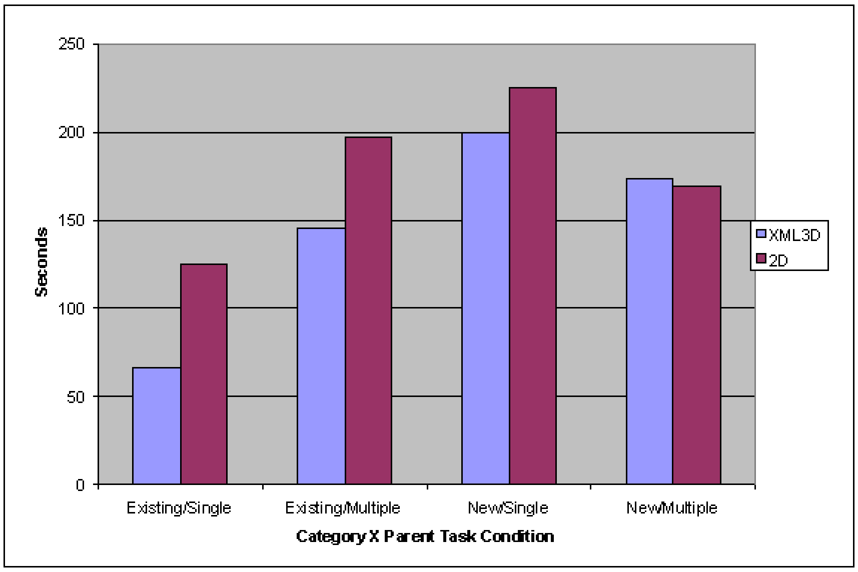

Figure 3.24: **XML3D vs. 2D interface study results.** In the user study we empirically compared the usability of XML3D to two established 2D interfaces: a collapsible tree browser such as appears in many Windows applications, and the click-through user interface of a Web portal. The two pairs of bars on the left side of the figure show the performance differences for the task of adding documents to an existing category, and the difference between the XML3D interface and the 2D interfaces is statistically significant. The two pairs of bars on the right side shows the performance differences for the task of creating an entirely new category in which to place a document, and these differences are not statistically significant. (Image courtesy of Mary Czerwinski, Microsoft Research.)

Figure 3.24 shows the task time results. We did not observe any reliable differences in task times between the two 2D browsers, so they are combined into a single category. The significant variable was the task type: adding content to existing categories vs. creating new categories. The results showed clear differences between XML3D and the 2D interfaces. With XML3D, participants performed search tasks within existing categories reliably faster with no decline in the quality of their responses. In the new category search task, there was no reliable overall difference. Interestingly, a breakdown of user activity logs indicates that in this no-improvement new category task case, users made little use of the XML3D interface component, spending most of their time interacting with the 2D list components. In the performance-improvement existing category task case, their time was roughly equally split between the interface components.

We can explain Figure 3.24 more formally: analysis of variance on mean task completion times revealed main effects of user interface condition (XML3D vs. 2D), $F(1, 11) = 10.19$, $p < .01$, and category (new vs. existing), $F(1, 11) = 37.76$, $p < .001$. Participants completed tasks faster using XML3D than using the more conventional interfaces. In addition, participants were faster to complete tasks involving an existing as

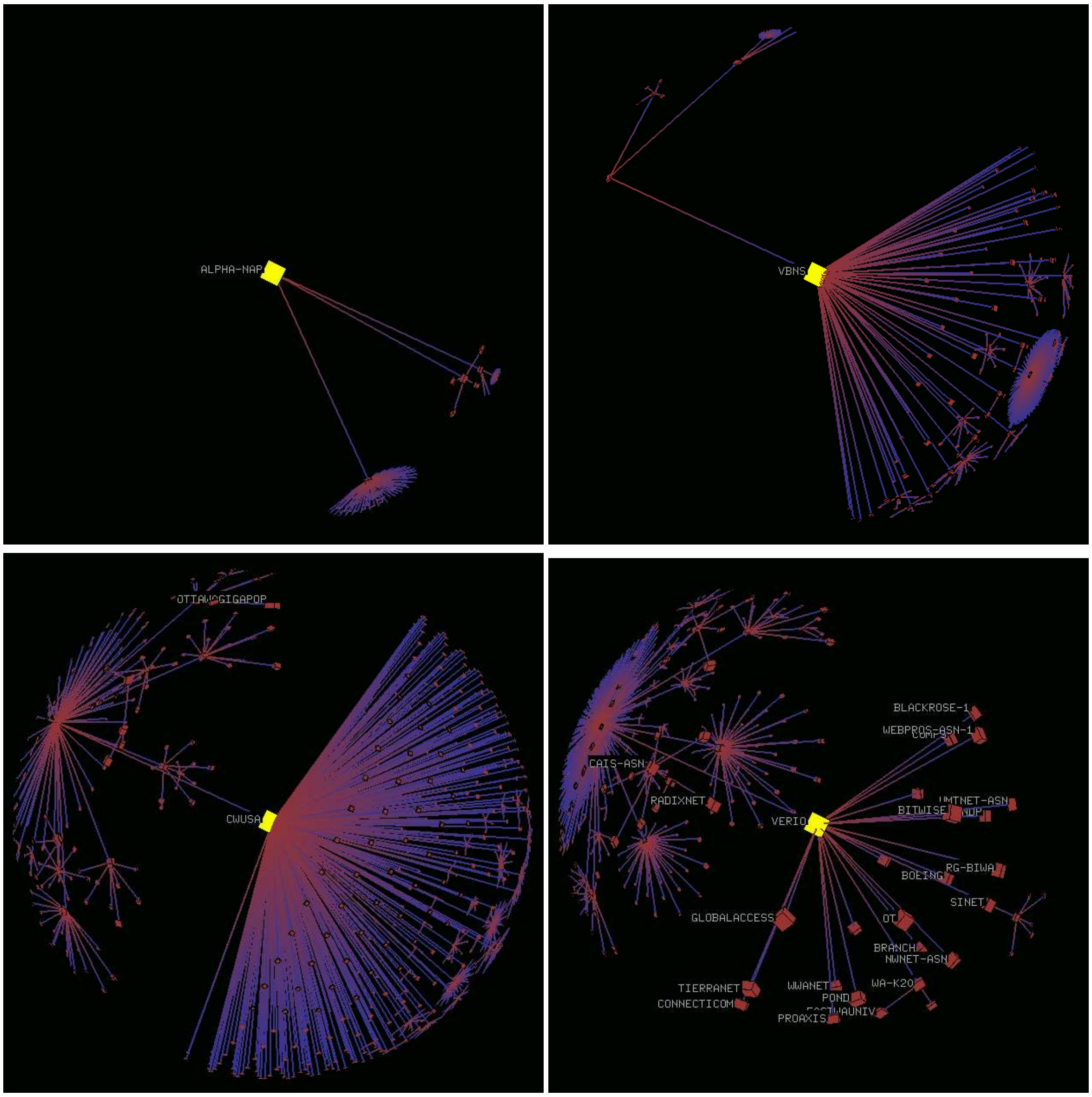

Figure 3.25: **Autonomous System paths analysis.** Images and (edited) caption courtesy of Daniel W. McRobb, CAIDA. The displayed spanning tree represents connectivity seen from `pinot.caida.org` on March 3, 1999. **Top left:** ALPHA-NAP is the root where we captured the AS path data. **Top right:** Here we clicked on the node at the center of the cluster in the bottom of our first view. Hypviewer brings the node to the front and center, and we see that this AS is the vBNS. **Bottom left:** Again following the large clusters, we clicked on the node at the center of the large cluster behind the vBNS, and found Cable and Wireless (CWUSA). **Bottom right:** Here we clicked on one of the small clusters off of CWUSA and found Verio. Since we're in a fairly small neighborhood, we see many labels.

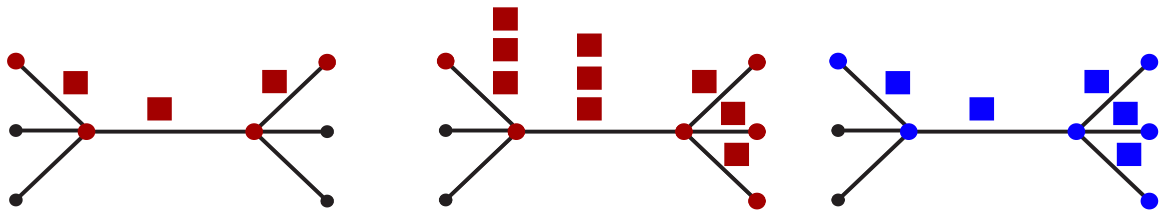

Figure 4.1: **Transport protocols. Left:** Unicast packet transport (shown here in red) was designed for one-to-one communication. **Middle:** Unicast packets sent from one source to multiple destinations results in congestion. **Right:** Multicast transport protocols allow packets (shown here in blue) to flow from one source to multiple destinations efficiently by dynamically maintaining a spanning tree of network links.

4.1 The Multicast Backbone Maintenance Task

The Planet Multicast visualization was aimed at helping the maintainers find badly configured parts of the MBone topology that wasted scarce resources.

4.1.1 Multicast

The MBone is the **multicast backbone** of the Internet. Multicasting is a network distribution methodology that efficiently transmits data from a single source to multiple receivers. The Internet routing protocols were originally designed to support **unicast** packets: that is, one-to-one communication from a single source to a single destination. The left of Figure 4.1 shows a simple example of unicast routing, which works well for applications such as email. However, these protocols are extremely inefficient when the same data is sent to many receivers, since the traffic load on the network increases linearly with the number of receivers, as in the middle of Figure 4.1.

The **multicast** routing protocol [CD85] dynamically manages a spanning tree over the links of its network. This tree ensures that identical data is sent only once across each multicast link, as shown on the right in Figure 4.1: data is replicated only as necessary when the spanning tree splits into multiple paths.

Such efficiency is important if there are a large number of destinations, or the data requires high bandwidth (for instance, video or audio streams), or both. The space shuttle launch video footage multicast by NASA to a large number of receivers is a canonical example. Many conferences, seminars, and other events are now viewable remotely thanks to the MBone, and it may see increasing usage as a communication mechanism for networked multi-user 3D applications.

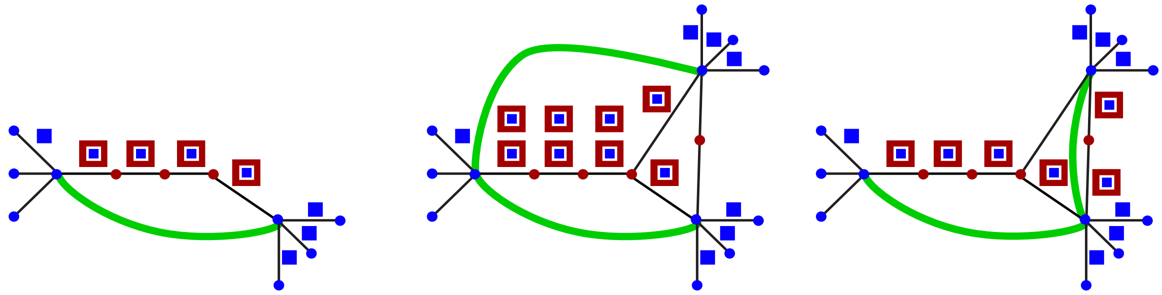

Figure 4.2: **Tunnelling. Left:** A tunnel (shown in green) allows new-style packets (blue) to traverse legacy infrastructure encapsulated in old-style packets (red), so that network improvements can be gradually deployed. **Middle:** A poor choice of tunnel placement, where identical packets traverse the same unicast link. **Right:** A more intelligent placement choice, where tunnels tie together islands of new infrastructure.

4.1.2 Tunnels

Adding new capabilities to the global Internet is difficult since simultaneously upgrading every single machine is not feasible. Since machines are upgraded piecemeal according to the schedule of local administrators, noncontiguous islands of new infrastructure must be connected by an overlay of virtual logical links, or **tunnels**, on top of the existing network. A schematic view of a tunnel is shown on the left of Figure 4.2. New-style data packets can move freely between machines that have been upgraded, but must be encapsulated at one end of the tunnel into old-style packets in order to be properly routed through the old infrastructure. These packets are unpacked at the other end of the tunnel, where the routers with the new capabilities can take over again.

Only some production Internet routers supported native multicast in 1996. The MBone was designed to provide interim support by using tunnels between routers that have been upgraded to support multicast, and those that are capable only of unicast routing. Multicast packets are encapsulated into normal Internet Protocol (IP) [Pos81] packets at an MBone tunnel endpoint.

4.1.3 Tunnel Placement

MBone tunnels are intended to stitch together the parts of the Internet that have not been upgraded to native multicast support. Tunnels should be placed so that they traverse the old infrastructure as little as possible, just long enough to get to the nearest available multicast router. Careful tunnel placement is important since the multicast protocol determines the shortest paths through its spanning tree based on the tunnel hop count rather than the underlying unicast hop count. The most efficient placement of a tunnel results in encapsulated multicast packets travelling on uncongested unicast links that do not have any other tunnels overlaid on top

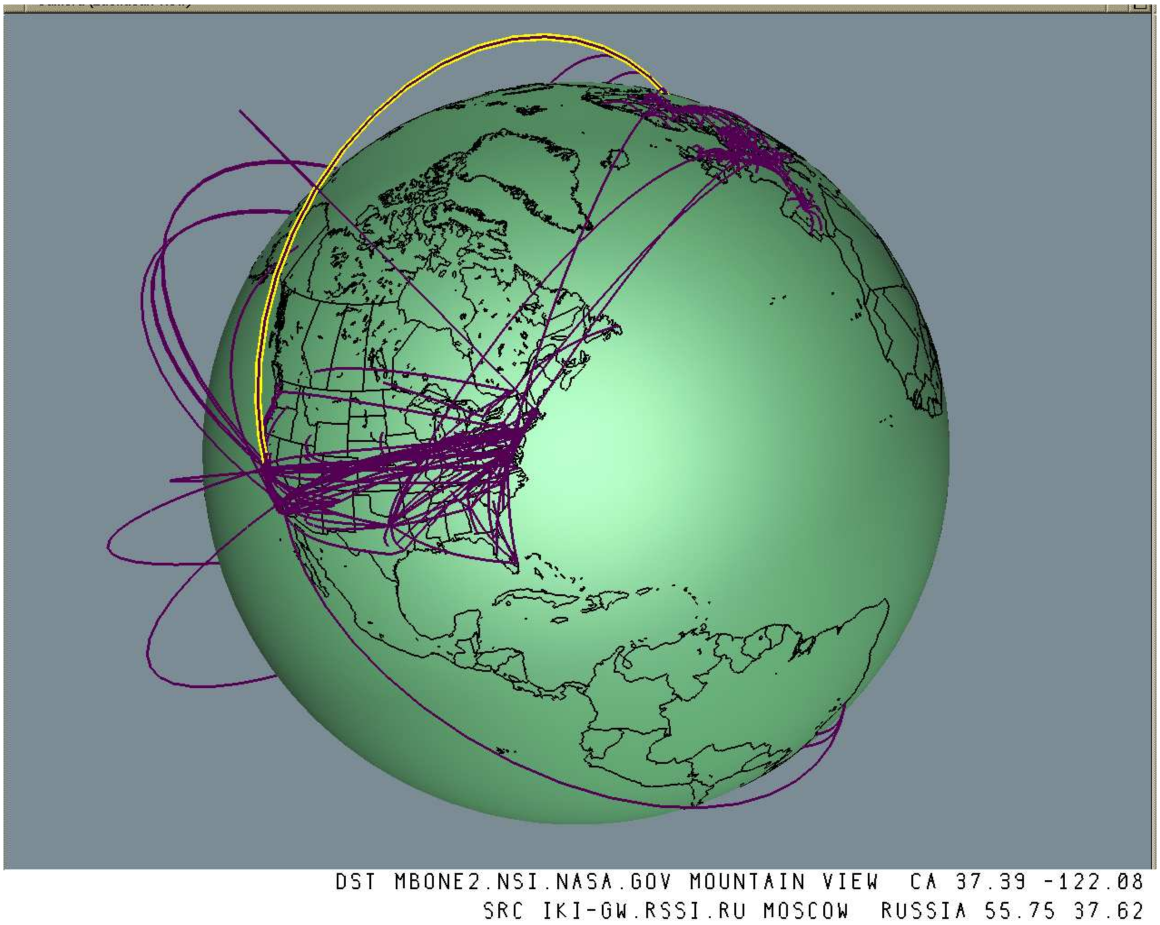

Figure 4.3: **MBone shown as arcs on a globe.** A 3D interactive map of the MBone tunnel structure, drawn as arcs on a globe. The endpoints of the tunnels are drawn at the determined geographic locations of the MBone router machines. In this and all figures not otherwise labeled, we draw nearly 700 of the 4400 tunnels comprising the MBone on June 15, 1996. Over 3200 are not drawn because we have determined the endpoints to be co-located, whereas the remaining 500 are ignored because we were unable to find their geographic position. The text window below shows the hostname, city and state or country name, latitude, and longitude of the endpoints of the highlighted tunnel, which was interactively selected by the user's click. A URL pointing to this information is bound to the 3D representation of each tunnel.

4.2.2 Implications of 3D Globe

There are two related implications of the arcs-on-globe visual metaphor choice: salience and filtering. The longest distance tunnels are deliberately the most visually salient. The inverse is also true: short-distance tunnels not at all visually salient (and indeed are not drawn at all). Tunnels that have both endpoints in the same city would be imperceptible on our global scale, therefore we draw tunnels only between two different cities.

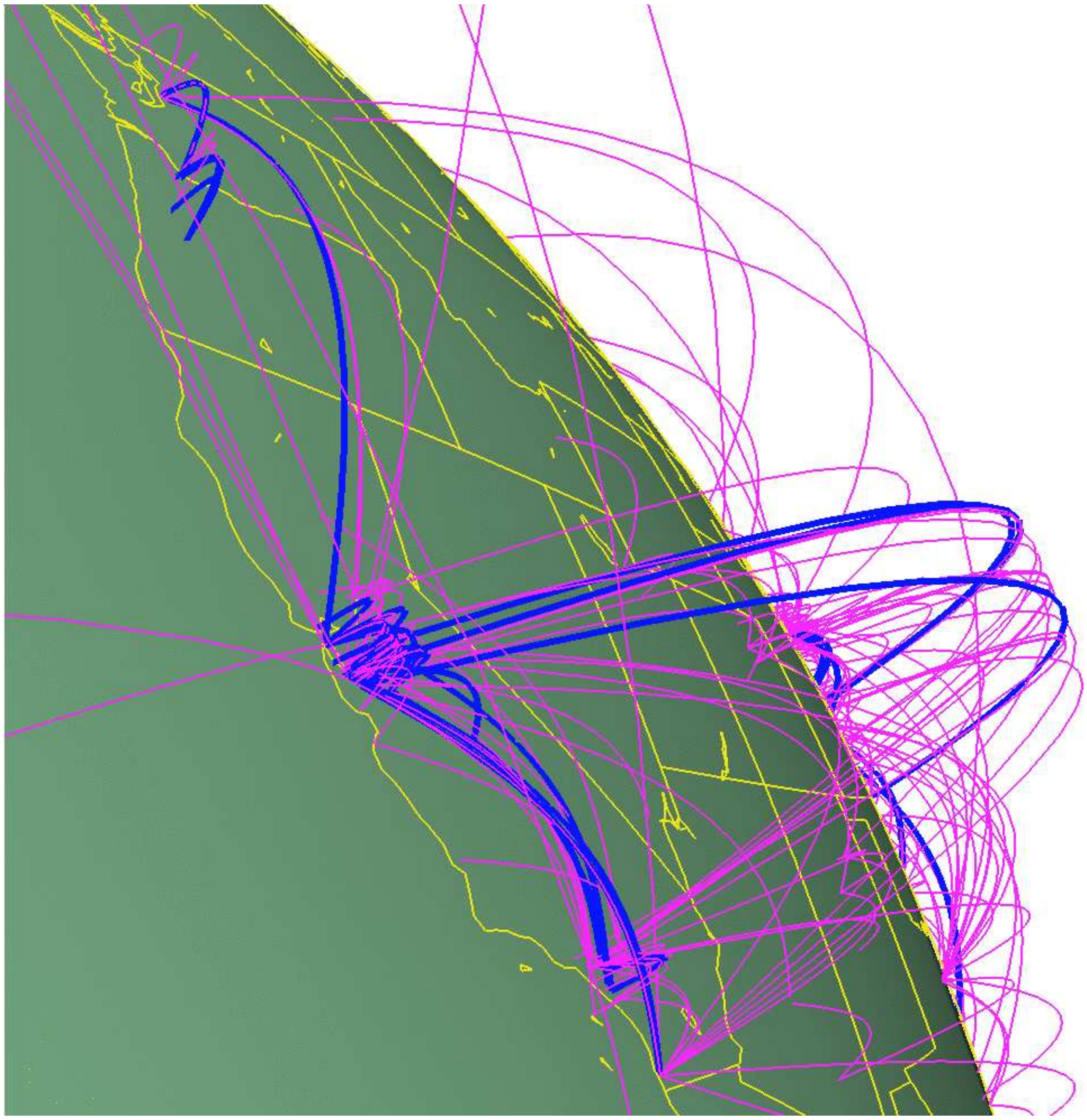

Figure 4.4: **Horizon view, arc height and grouping.** Three visualization techniques are used in this figure: distance-based arc height, 3D navigation and grouping. The "horizon view" results from zooming close and clicking on a point on the earth's surface to act as the center of rotation. Moving our eyepoint close to the earth also emphasizes the different arc heights. Finally, the group of tunnels running the PIM protocol are drawn with a different color and linewidth than the rest.

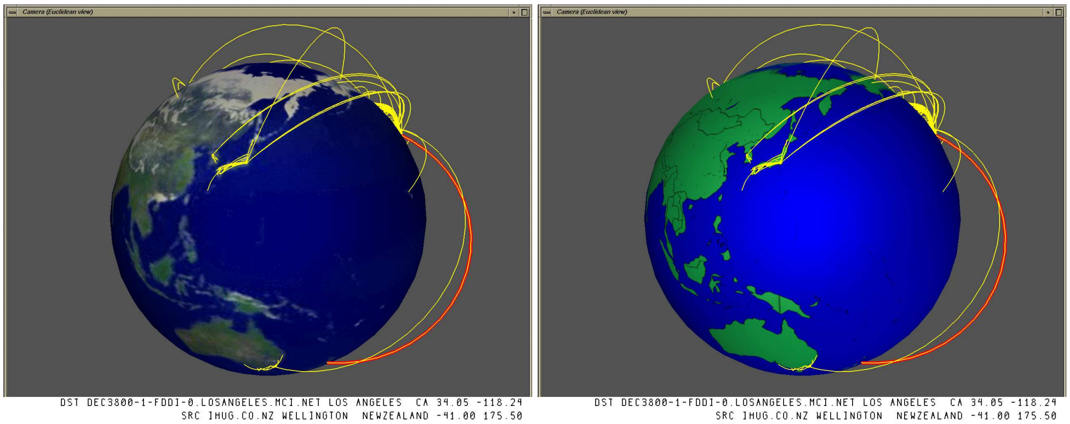

Figure 4.5: **Textured globes. Left:** A more photorealistic texture map offers a familiar global picture but introduces extraneous visual clutter and requires more computational resources. **Right:** A more abstract texture is still too computationally intensive for low-end platforms, but it is easier to distinguish land from water than on a globe with only vectors for continental outlines.

4.2.3 Spherical Geodesics

Computing the spatial position of arcs on a globe reduces to 2D spherical trigonometry. We convert the latitude and longitude of the two tunnel endpoints into spherical coordinates (ϕ, θ), and find the shortest geodesic arc on the surface of a unit sphere between those two two points. We then loft the geodesic to a maximum height h that depends on its length. The arc geometry is created as a controllable number of piecewise linear segments.

We use the same equation as the SeeNet3D system [CE95] for computing the lofted height of the arc:

$$R = 1 + h \sin(\pi t), 0 \leq t \leq 1 \tag{4.1}$$

The parameter t ranges from 0 to 1 along the arc path.

We briefly experimented with using the same height for all arcs, but the display quickly became overly cluttered. Having the arc height depend on its length lends visual emphasis to long arcs, an advantage for our application since we want long-distance tunnels to stand out. Likewise, short tunnels are less obvious, which is appropriate since such tunnels are assumed to have less effect on global congestion. We do impose a minimum arc height requirement so that even the shortest tunnels remain visible. Variable arc height is a visual encoding that is useful only with a three-dimensional visual metaphor, since it would not be meaningful if used with a 2D birdseye view. Figures 4.4 and 4.9 (page 84) show how an oblique viewpoint close to the surface of the globe makes the varying arc heights more visible.

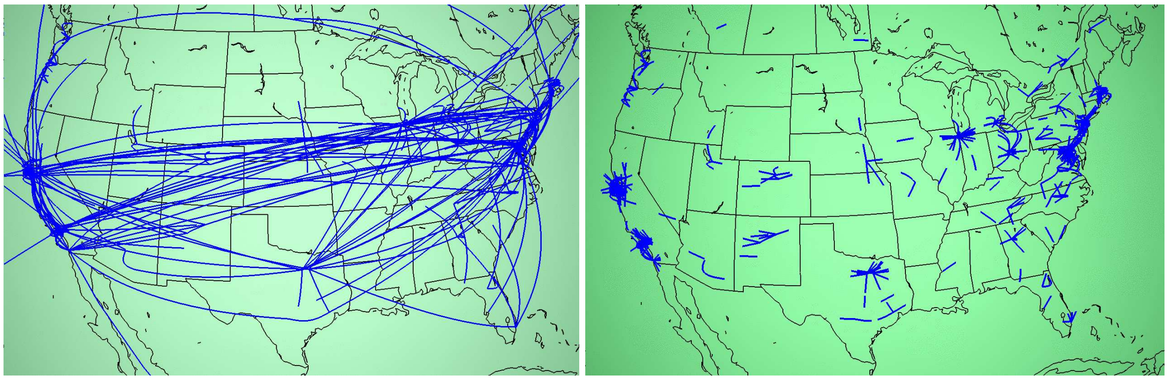

Figure 4.6: **Thresholding. Left:** Long-distance tunnels that cross from coast to coast in the US obscure local endpoint details in the Midwest. **Right** Only the segments within a user-defined radius around the tunnel endpoints are drawn in this thresholded view, which reveals local details in the middle of the country.

lightweight pipeline of batch modules.[5] The pipeline phases were:

- canonicalize the data into name-value pair format
- resolve hostnames and IP addresses into geographic locations
- group sets of tunnels according to name-value filters
- construct piece-wise linear arc geometry.[6]

No geometry for a tunnel is created in cases where we were unable to resolve a location for both its endpoints.

4.4.1 Geographical Determination

In order to construct the geographical representation, we need to obtain the latitude and longitude that corresponds to the IP address of each MBone router. A database maintained by InterNIC contains a geographic location for every Internet domain. Unfortunately, this information is useful only for domains with a single physical location, such as a university campus. A single contact address is not helpful for large companies with many branches, or worse yet an entire network. Even non-backbone domains such as `nist.gov` or `csiro.au` can encompass several different campuses within an organization. Since by nature many important MBone nodes belong to transit providers that cover a wide geographic area, the InterNIC-registered

[5] Parts of this pipeline were implemented by Eric Hoffman.

[6] We used previously existing spherical geodesic code extracted from the *spherescribble* interactive software by Millie Niss, available from `http://www.geom.umn.edu/software/download/spherescribble.html`.

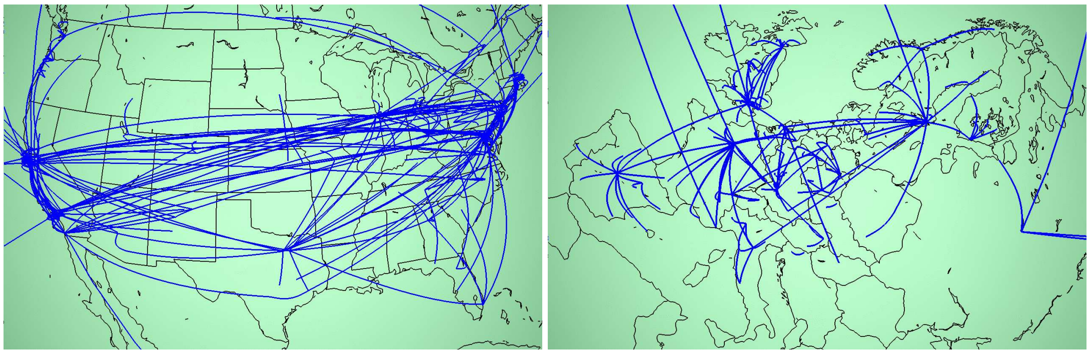

Figure 4.7: **Two regional closeup views of the MBone. Left:** United States tunnel structure. **Right:** European tunnel structure. The US tunnel structure is quite redundant compared to the European one. Reducing the number of coast-to-coast tunnels would reduce the offered workload to the often congested underlying unicast infrastructure. Many of these tunnels may be carrying identical data.

4.5 Results

We believed that disseminating 3D data files would allow maintainers to interactively explore the MBone structure and gain clearer understanding of the problems and possible solutions than would be available from still pictures or even videos. The Planet Multicast visualization software did provide one MBone maintainer with some insights into the topology of the MBone in 1996, and was used in a few additional networking visualization task domains. However, the system was never fully deployed because of scalability problems with geographic determination, and it is not currently in active use by its target audience.

4.5.1 Topology Insights

The most obvious conclusion about the general character of MBone deployment in 1996 was that areas with fewer network resources and limited numbers of redundant links seem to have more efficient tunnel placements. The European tunnel structure shown in Figure 4.7 is much closer to a hierarchical distribution tree than that of the United States. The commercialization of the US-based Internet in the mid-1990's led to the fragmentation of the formerly hierarchical US structure.

The US topology seemed to be highly nonoptimal at first glance. Although the MBone maintainers knew that there were some redundant tunnels, the sheer number of coast-to-coast tunnels visible with the Planet Multicast display was surprising even to them. We split the tunnels into groups according to the major Internet Service Provider (ISP) backbones to investigate further. Figure 4.8 shows the tunnels partitioned into three sets: both tunnel endpoints belong to a major backbone (black), one endpoint is on a backbone and the other is not (blue), or neither endpoint connects directly to a backbone (red). We learned two lessons from the color

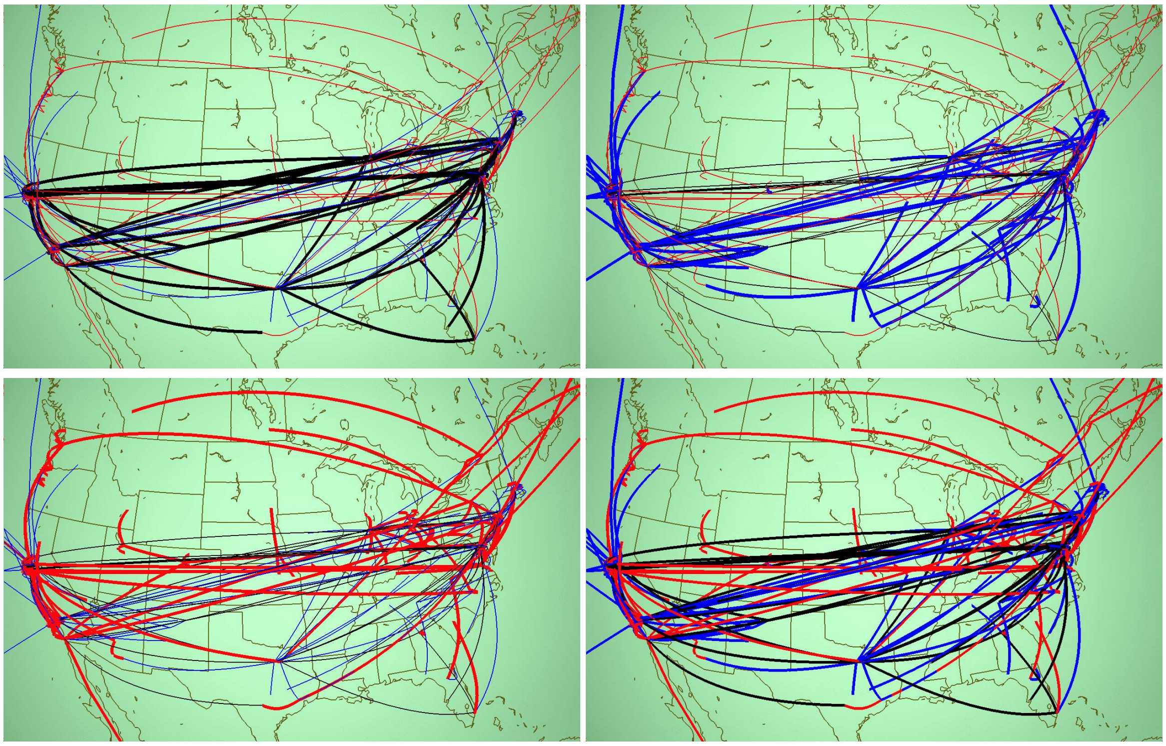

Figure 4.8: **MBone tunnels grouped by backbone status.** Here we show the United States in June 1996, which is the same data as the right side of Figure 4.7. Groups are color-coded according to whether tunnel endpoints are belong to a major backbone Internet Service Provider. We sought to understand whether the profusion of coast-to-coast tunnels was excessively redundant or a reasonable consequence of the commercialization of the US backbone. **Top Left:** Black tunnels, which have a major provider at both ends, are emphasized. If most of the long tunnels were black, then the redundancy might be quite reasonable. **Top Right:** Blue tunnels, which have one endpoint on a backbone and one endpoint in a non-backbone domain, are emphasized. These tunnels are possibly legitimate, since ISP network administrators are presumably more motivated to make sure that their bandwidth is not being wasted than an average corporate or university sysadmin. **Bottom Left:** Red tunnels, which have neither endpoint on a major backbone, are emphasized. These are prime targets for suspicion, and there are a distressingly large number of them. **Bottom Right:** All three groups are equally emphasized with the same line weight, so that their relative sizes can be compared visually.

In another case, the Planet Multicast toolkit was used by Andrew Hoag of NASA-Ames to show the geographic structure of the emerging 6Bone.[11] The 6Bone, like the MBone, is way to deploy a new network service gradually through tunnels. In this case, the service is IPv6 [DH98], the new version of the Internet Protocol. Hoag started using our toolkit to create his own page in late 1996, but since he is no longer at Ames the interactive 3D maps are no longer current.

[11]`http://www.nas.nasa.gov/Groups/LAN/IPv6/viz/`

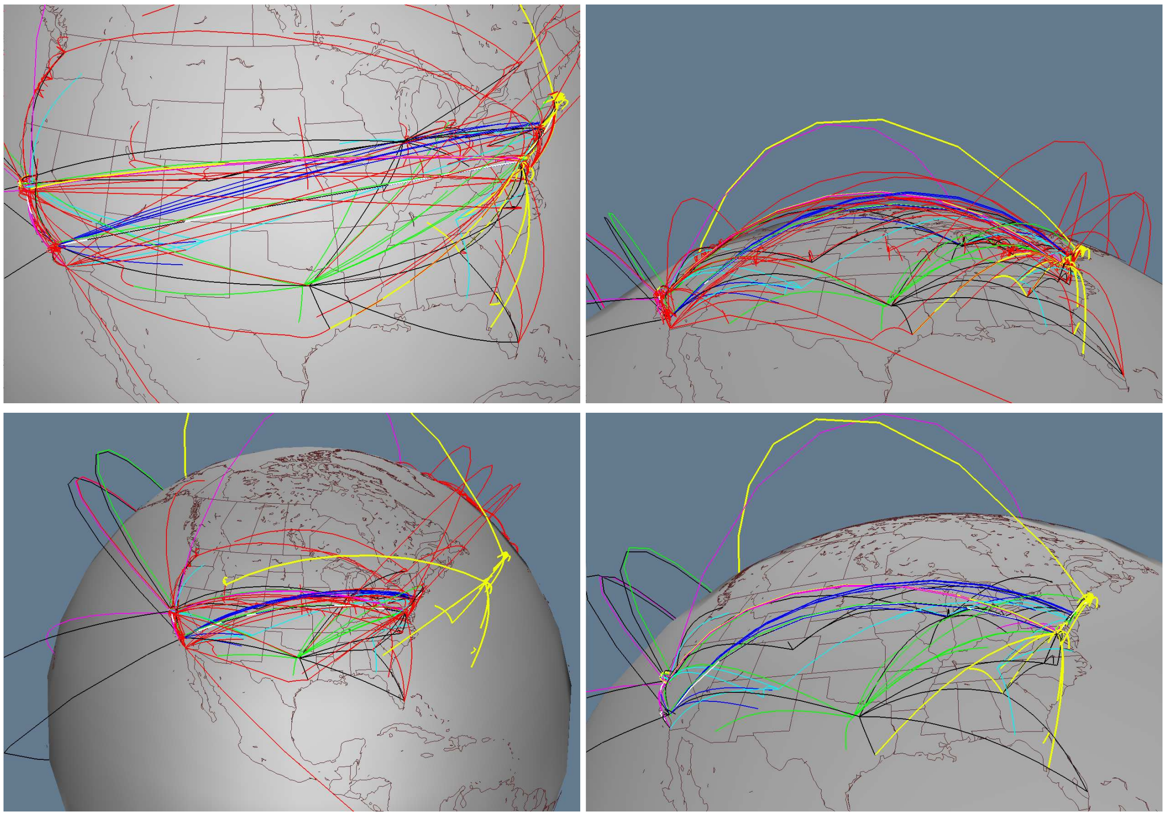

Figure 4.9: **MBone tunnels of the major backbone networks, colored by provider.** We show the same data (United States, June 1996) as Figure 4.8, but drill down further by grouping according to the backbones themselves to check that no individual backbone has excessive redundancy. The tunnel color coding is black for MCI, green for Sprintlink, blue for ANS, cyan for ESnet, magenta for NASA, yellow for BBNPlanet, and white for Dartnet. These tunnels correspond to the groups colored black and blue in Figure 4.8. Tunnels that are not connected to backbones are colored in red in both that figure and this one. **Top Left:** Everything from a birdseye view. **Top Right:** A horizon view takes advantage of the varying arc height to make the structure more obvious. **Bottom Left:** We move the BBN tunnels away from the main mass to see them more clearly, and the aggregate bicoastal structure is still obvious even though they are no longer anchored to their true geographic location. **Bottom Right:** We have interactively elided the red non-backbone tunnels, showing that each individual backbone has a reasonable structure when considered alone.

4.5.3 Outcomes

The deployment of a system such as the MBone should be a careful balance between distribution efficiency, resource availability, redundancy in case of failure, and administrative policy. Our hope was that this visualization system would help ISPs and administrators of campus networks cope with a growing infrastructure by illustrating where optimizations or more appropriate redundancies could occur within and across network

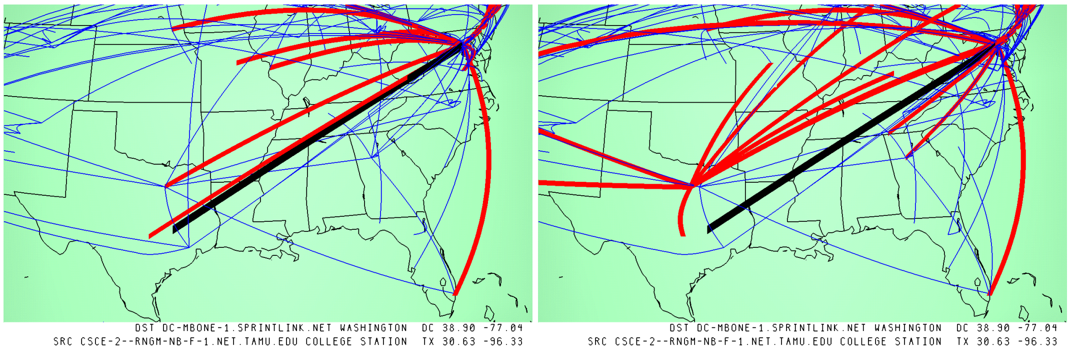

Figure 4.10: **MBone tunnel structure in Texas at two different times. Left:** February 12, 1996. **Right:** June 15, 1996. Tunnels in the Sprintlink network are drawn in thicker red arcs in both, and the selection of a tunnel between Texas A&M University and Washington, DC is shown with a thick black arc. In the later picture we see that although Sprint has established a major new hub close to TAMU, that closer source is still unleveraged in June.

boundaries. When Bill Fenner, who is heavily involved with MBone deployment, saw the initial 3D visualization in February 1996, he was galvanized to encourage many administrators of suboptimal tunnels to improve their configuration. Although he was familiar with the textual data, the geographic visualization highlighted specific problems in the distribution framework that he had not previously noticed.

The 3D visualizations were primarily intended for people working in the MBone engineering process because their interpretation requires a great deal of operational context. Although they could be misleading if seen as standalone images by those without a broad understanding of the underlying technologies, they can serve as an educational medium for the general public with appropriate interpretation.

We hoped that these visualizations would encourage network providers to make available geographic, topological, and performance information for use in visualizations that could facilitate Internet engineering on large and small scales. Although our toolkit was also used for a few other networking visualization projects, it is not currently in active use by MBone maintainers. In the next section we discuss the barriers to adoption that prevented its deployment to the MBone maintenance community. We thus have no data about the response of our intended users to the visual metaphor.

4.5.4 Barriers to Adoption

The main stumbling block to widespread adoption was the nonscalability of our necessarily ad-hoc geographic determination techniques. Although it was possible to glean most of the necessary information for the MBone circa 1996, these methods were infeasible for the larger Internet or even the MBone after 1997. We made our database publicly available in hopes that it would become self-sustaining through enlightened

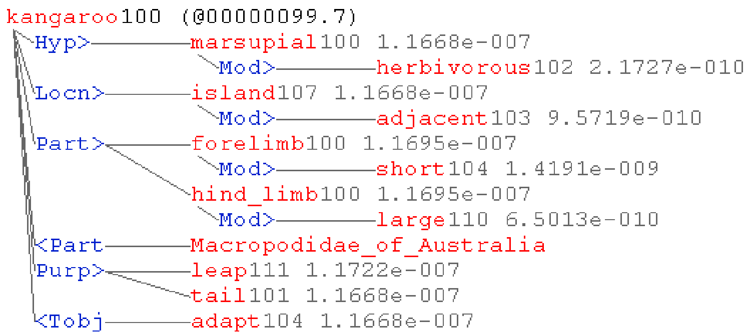

Figure 5.1: **Parsed definition graph from MindNet.**The parsed definition graph for KANGAROO100 is a mixture of plausible attachments, such as KANGAROO `is-a` MARSUPIAL, and errors, such as the misattachment of AUSTRALIA to the Latin name for the species instead of to the phrase ADJACENT ISLANDS.

5.1.2 Plausibility-Checking Task

Both the target users and the author agreed that the primary goal was an exploratory system that would be actively useful in day-to-day research. The possible goal of an expository demonstration was deemed to be less important.

Although MindNet is extremely successful by the standards of the NLP field, it is known to be imperfect. The semantic network is automatically constructed, but a feedback loop is part of their ongoing research program: the answers returned by MindNet are hand-checked by human linguists, who determine whether they are plausible. Problems are addressed when possible by improving the algorithms used to create MindNet, and the network is regenerated.

When we first heard about the plausibility-checking task in the initial interviews, we thought that the researchers would want to see a global overview of the entire network, and anticipated tackling some large dataset algorithms. However, further discussions soon revealed that a large-scale global overview was not what the researchers needed. They already understood the major features of the global dataset, which is highly connected: one word can connect to most of the other words in the network after three or four hops, and to all in five. The semantic networks generated by MindNet are sufficiently large and interconnected that its developers find it impractical to study their global structure for plausibility-checking purposes.

They instead rely on a query engine to probe a small subsection of the network, and each of these snapshots is checked for potential problems. The linguist user provides a query consisting of two words and the number of **paths** to return. MindNet computes the best paths between the words, as shown in Figure 5.2. The system returns the requested number of paths in order according of computed **plausibility**, which is derived using (among other factors) the edge weights in its unified network of definition graphs. Each path is

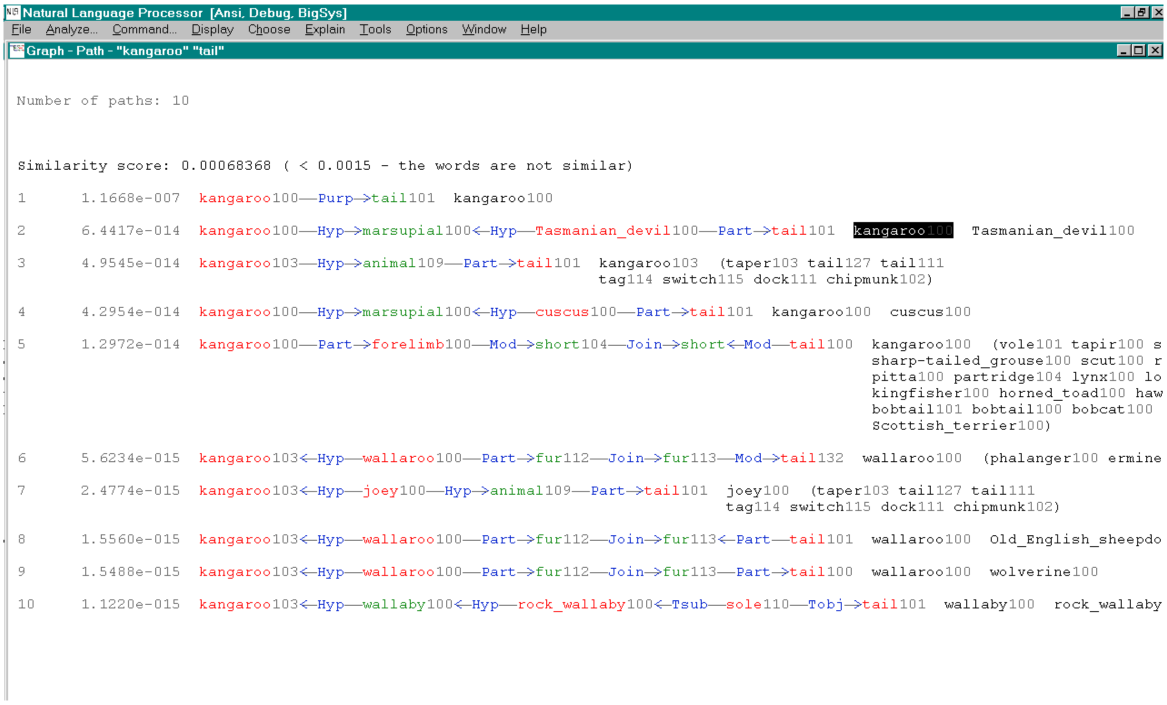

Figure 5.2: **Previously existing textual view in MindNet.** The query results returned by MindNet when asked for the ten best paths between KANGAROO and TAIL. On the left are the colored words in the path itself, and on the right in black are the first words in each of the definition graphs used in the computation. The first path is only one hop since TAIL101 is present in the definition of KANGAROO100. That word is highlighted in black because the user has clicked on it, triggering a popup window showing the information in Figure 5.1. Making plausibility judgements in this interface requires a great deal of flipping between windows. The second path uses the definitions for both KANGAROO100 and TASMANIAN_DEVIL100 (shown in Figure 5.10 on page 100), both of which contain the word MARSUPIAL100. The fifth path is an example of one that required a large number of definition graphs to compute.

accompanied by the first words of every definition graph used in its computation. These words are shown in black on the right side of Figure 5.2. The linguist would hand-check the results to see how well the computed plausibility matched the intuitions of a human: for example, that all high-ranking paths were believable, and that all believable paths were highly ranked. Another check was for stop words that might be polluting the dataset: that is, words such as SHE, IT or THE, which are so common in English that they are usually excluded from computations.

A single word sense can appear in multiple places in a query result: for example, KANGAROO103 is included in two different paths, appears as a **leafword** inside the definition of WALLABY100, and also appears as a **headword** with its own definition graph. These shared words are the reason it is difficult to understand how paths relate to each other and to the definition graphs used to create them.

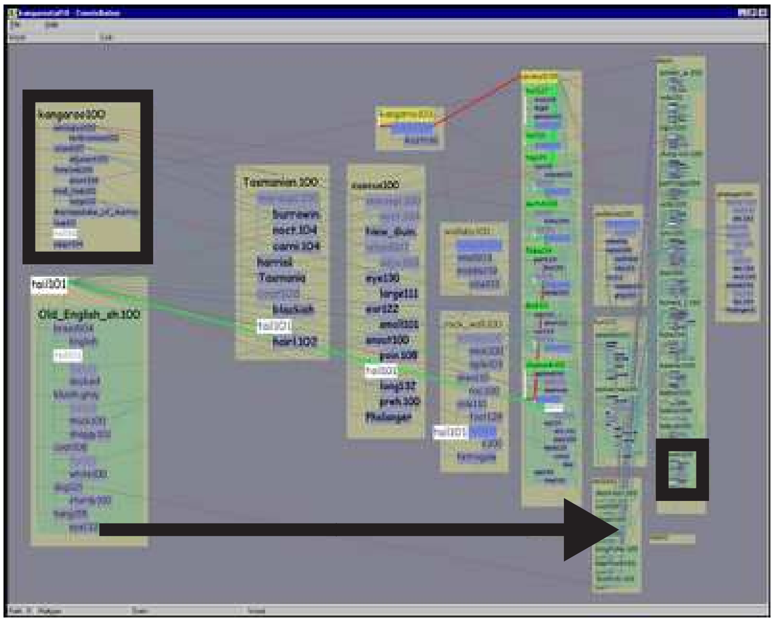

Figure 5.4: **Plausibility gradient encodes a domain-specific attribute.** Since spatial position is the strongest perceptual channel, we use it to communicate information about the domain instead of devoting this channel to avoiding the problems of false attachments because of edge crossings. The horizontal position of a definition graph is tied to MindNet's computed plausibility, and boxes on the plausible left are drawn larger than those on the implausible right. We avoid false attachments using selective highlighting, as shown in Figure 5.5.

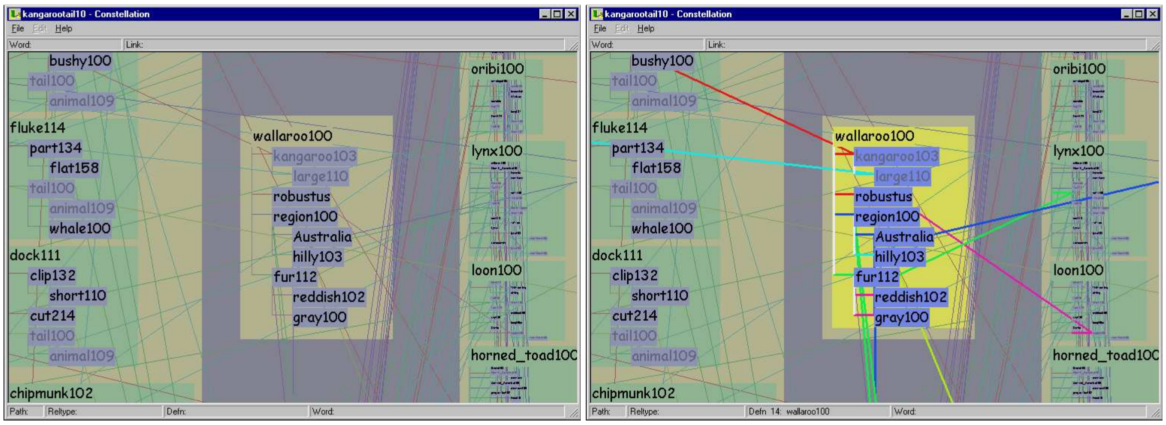

Figure 5.5: **Selective emphasis avoids perception of false attachments. Left:** Our layout algorithm results in crossings for the long-distance edges that connect all instances of a shared word. **Right:** Selective emphasis through interaction and additional perceptual channels avoids the *perception* of false attachments.

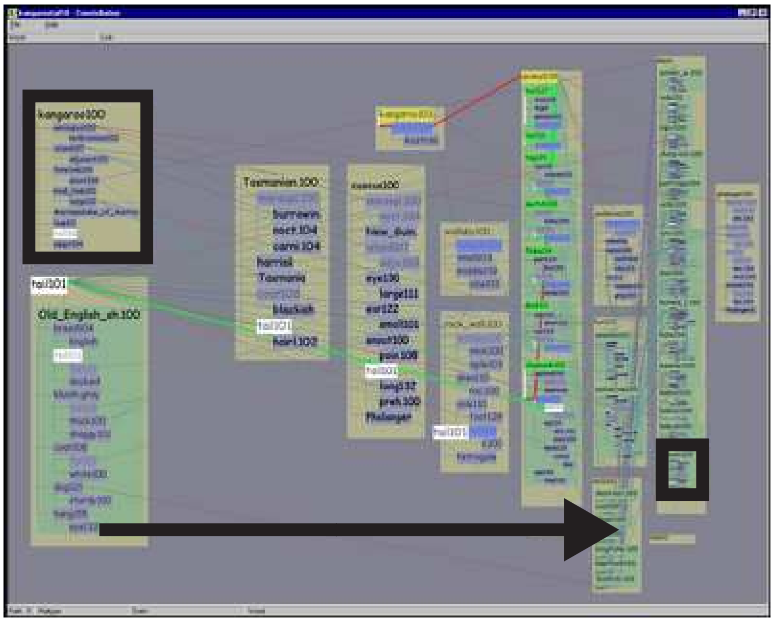

Figure 5.4: **Plausibility gradient encodes a domain-specific attribute.** Since spatial position is the strongest perceptual channel, we use it to communicate information about the domain instead of devoting this channel to avoiding the problems of false attachments because of edge crossings. The horizontal position of a definition graph is tied to MindNet's computed plausibility, and boxes on the plausible left are drawn larger than those on the implausible right. We avoid false attachments using selective highlighting, as shown in Figure 5.5.

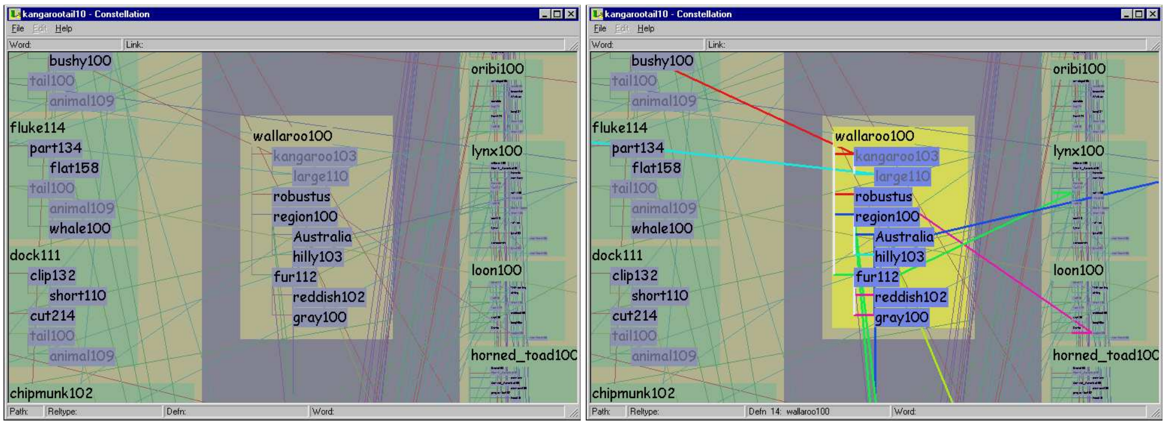

Figure 5.5: **Selective emphasis avoids perception of false attachments. Left:** Our layout algorithm results in crossings for the long-distance edges that connect all instances of a shared word. **Right:** Selective emphasis through interaction and additional perceptual channels avoids the *perception* of false attachments.

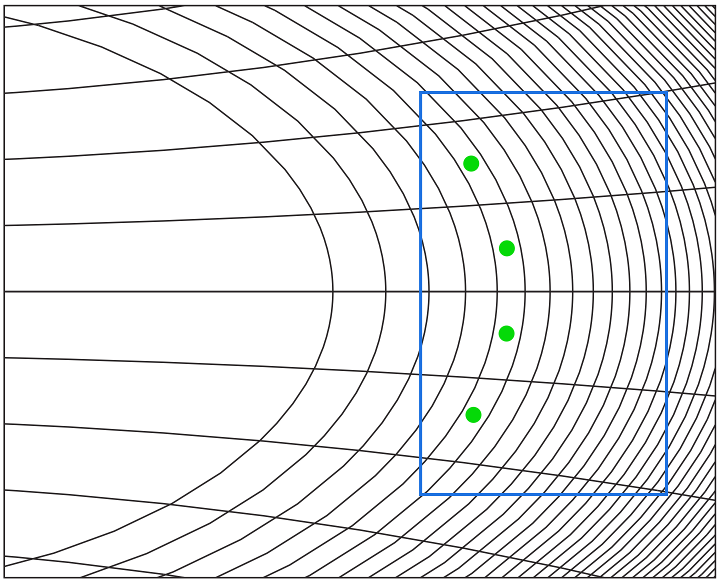

Figure 5.8: **Curvilinear grid.** Parabolas and circles are used to generate the curvilinear grid that is the basis of our layout algorithm. Here the aspect ratio of the display elongates the circles to give them the appearance of ellipses. The distance between circle radii decreases on the implausible right. The section of the grid used in a typical figure is denoted by the blue box, with green dots indicating the first band of cells used. The difference in curvature between the grid shown here and the grids visible in Figure 5.13 (page 104) is a result of fitting the blue box to the aspect ratio of the display window.

family of parabolas with a family of circles. We wanted bands on the implausible right to be thinner than those on the plausible left, so that the circle radii decrease logarithmically according to the horizontal plausibility gradient.

We parametrize a parabola for row i as

$$y = f_i + g_i x^2 \tag{5.1}$$

After experimentation, we empirically set the height offset f_i of the parabola simply to i, and the "curviness" g_i to $\frac{1}{200}$. Solving for x gives us

$$x = \sqrt{\frac{(y-f)}{g}} \tag{5.2}$$

The circles for column j are parametrized by

$$x^2 + y^2 = r_j^2 \tag{5.3}$$

Figure 5.9: **Attaching definitions to path segments.** Every definition graph is drawn attached to a pathword. **Left:** When a pathword has been assigned its own definition graph, it is drawn at the top of the path segment box against a tan background instead of nested in the usual green box. **Right:** Some words appear on a path because they appear as a leafword in a definition graph, and are not themselves defined. In this case only the pathword is drawn against the tan background, and each attached definition graph is drawn in its green box.

two green-boxed definition graphs associated with it. Some path computations involve the pooled influence of many definition graphs for a single pathword, so there may be many green boxes vertically stacked inside a single tan pathword box, as in the top left of Figure 5.14 on page 108 . Path 7 of the KANGAROO-TAIL ten-path dataset is an extreme example in the bottom right of that figure, and is also visible as text in Figure 5.2 (page 90).

5.2.6 Definition Graph Layout

Definition graphs are drawn with a ladder-like rectilinear structure, showcased in Figure 5.10, that is deliberately similar to the layout familiar to the linguists (shown in Figure 5.1 on page 89). Each leafword is enclosed in its own blue label box. Vertical edges show the hierarchical microstructure inside the definition graph, and horizontal edges are color coded to show the relation type. (The full list of colors is given in Table 5.1 on page 113.) The left and middle of Figure 5.10 show the highlighted state, and the right the unemphasized state.

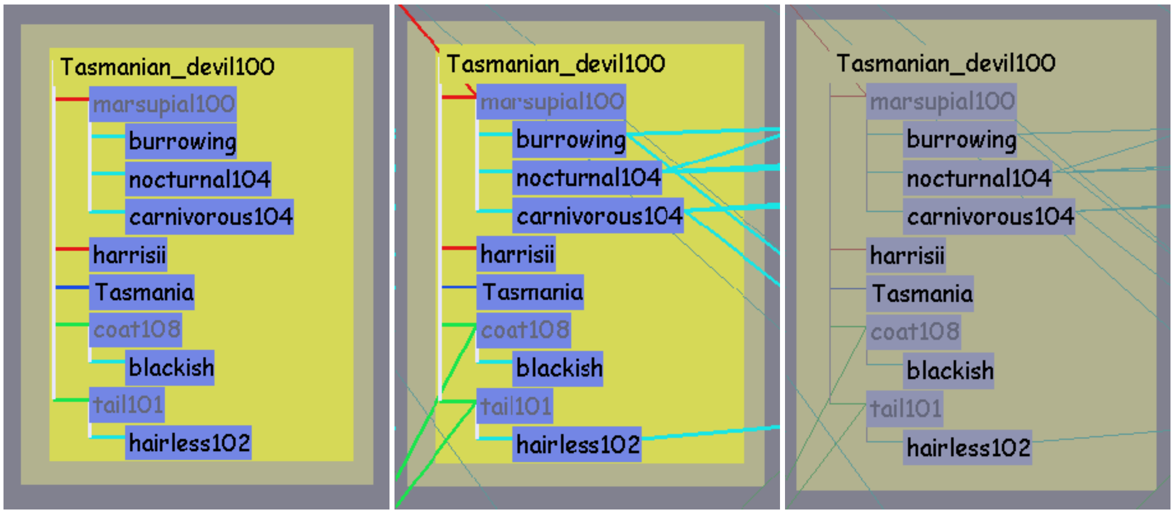

Figure 5.10: **Definition graph layout. Left:** In the base layout, words are connected by rectilinear links. The headword is drawn at the top left and leaf words are enclosed in blue boxes. All vertical lines are white, and the horizontal lines are colored according to relation type. **Middle:** We draw long-distance links between the master version of a word and all its duplicated proxies. **Right:** The unhighlighted state is the default when a definition graph is not the focus of the user's attention.

Pathwords that are shared among many paths are combined to create a semantically meaningful global structure encoding computed plausibility. Although pathwords and thus entire definition graphs are drawn only once, a word that appears in more than one definition graph will be drawn multiple times. We designate the **master** version of a definition graph leafword to be the one attached to the most plausible path, and draw it in black. All subsequent instances of the word are **proxy** versions, which are drawn in grey and visually connected to the master word by a long slanted line. These lines are visible in the middle and right sides of Figure 5.10, and in all other Constellation screen shots. The left side of Figure 5.10 is the only figure that does not show the long-distance proxy links, so as to showcase the base rectilinear definition graph layout.

An earlier iteration of our layout drew only the master word, leaving the proxy slots empty, as in Figure 5.11. Our intent was to ensure that the users were aware that proxy words appeared in multiple places in the input data. However, we observed that the linguists spend a significant amount of time reading individual definition graphs, and were disoriented when forced to navigate back along the proxy links during closeup reading. In the final version we instead optimized the spatial layout to support this reading task. In Section 5.3 we discuss the interaction paradigm that we designed to ensure that the connections between multiple instances of a shared leafword were always visually salient.

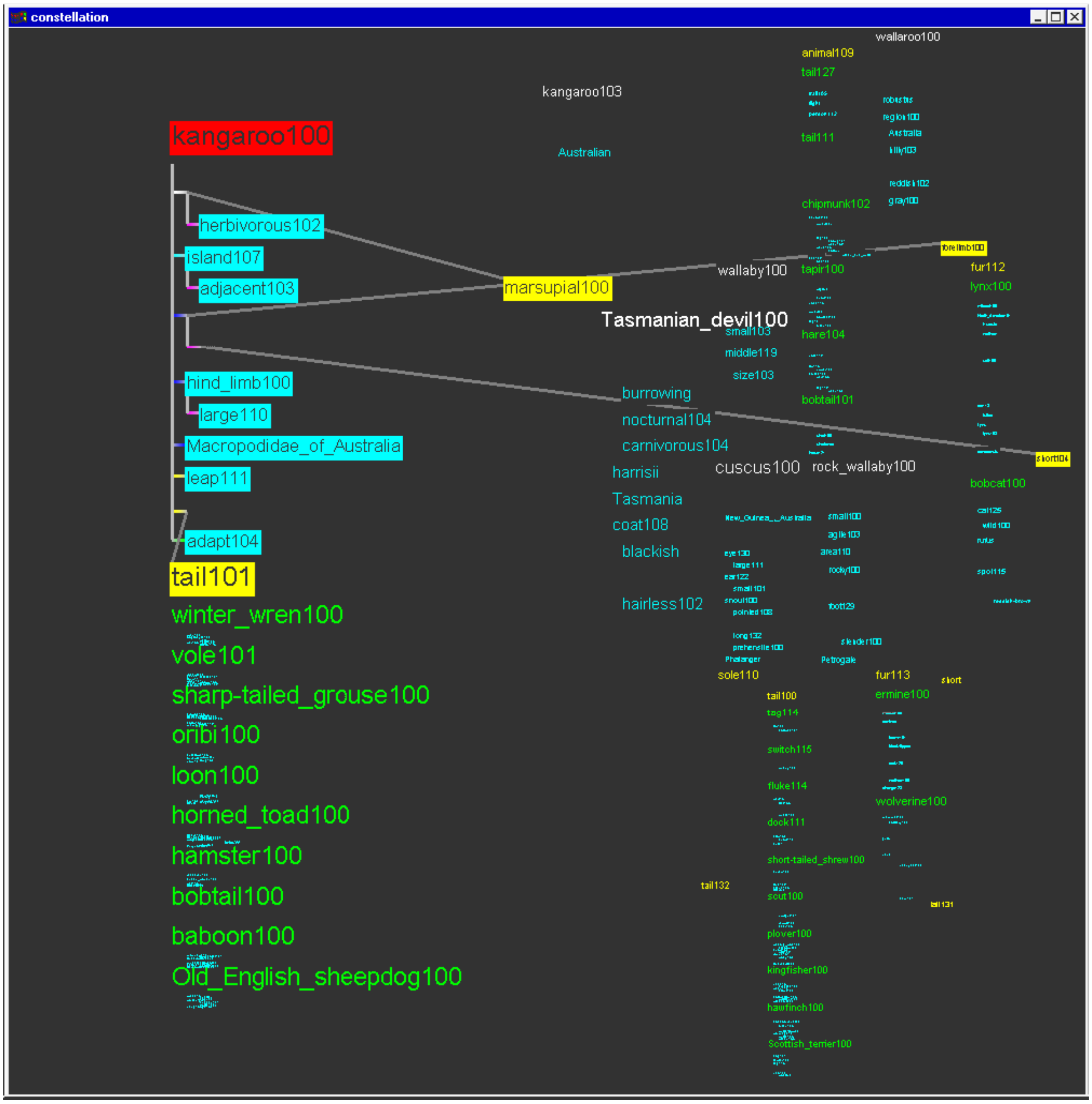

Figure 5.11: **Very early layout with empty proxy slots.** In this very early layout attempt only master versions of words were drawn, so it was hard to read any single definition graph when zoomed in. Here the definition graph constellation for KANGAROO is highlighted.

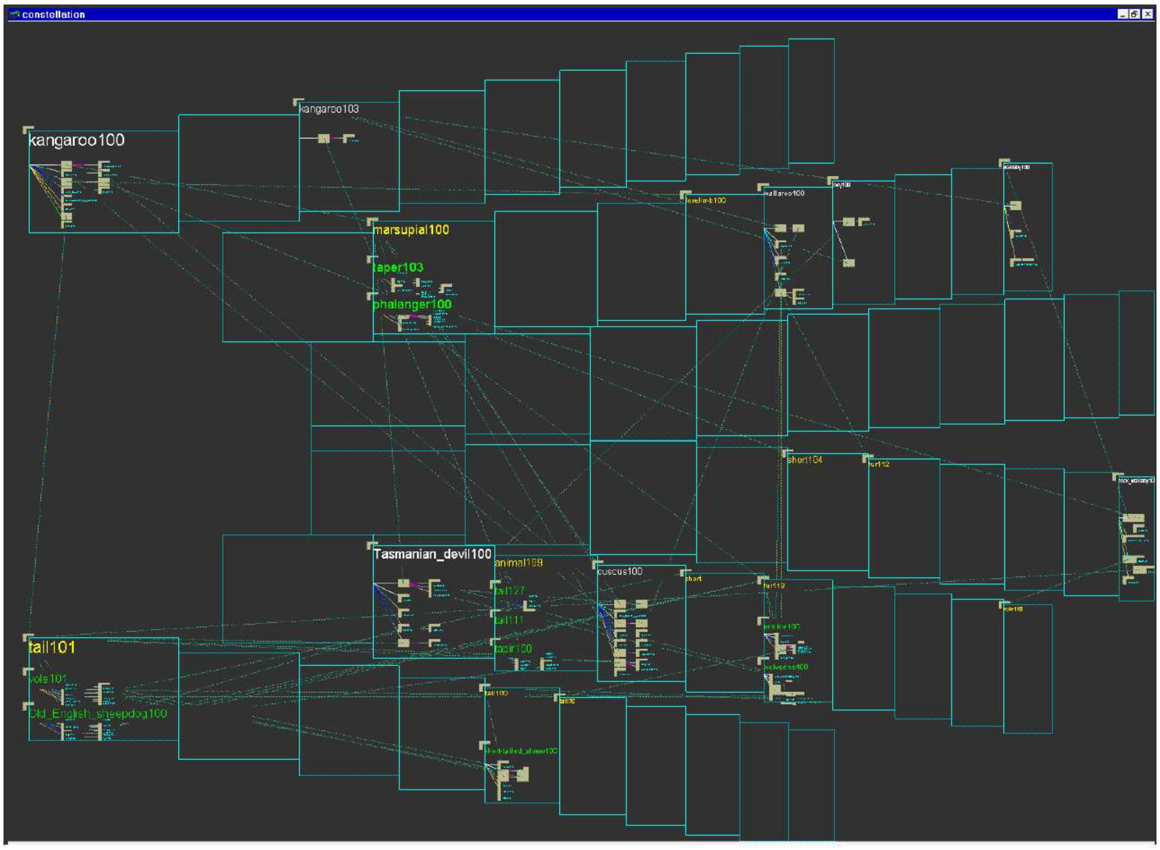

Figure 5.12: **Early sparse layout.** An earlier version of the software used only the base layout algorithm described in the previous sections, which is successful at encoding the plausibility spatially but results in a somewhat sparse layout with only about 20 legible words.

5.2.7 Increasing the Layout Density

We need to balance the two competing needs of creating a spatial arrangement that faithfully reflects the structure of the dataset, and filling space to achieve a uniform information density. Figure 5.12 shows a layout from one of the earlier software prototypes, with a grid constructed according to the base algorithm described in section 5.2.2. The layout exactly represents the desired domain specific information, but is quite sparse. Although the empty space does have meaning, we can achieve much greater information density. Figure 5.13 shows a progression towards a more dense layout that retains almost all the informative semantics of the base algorithm shown in Figure 5.12.

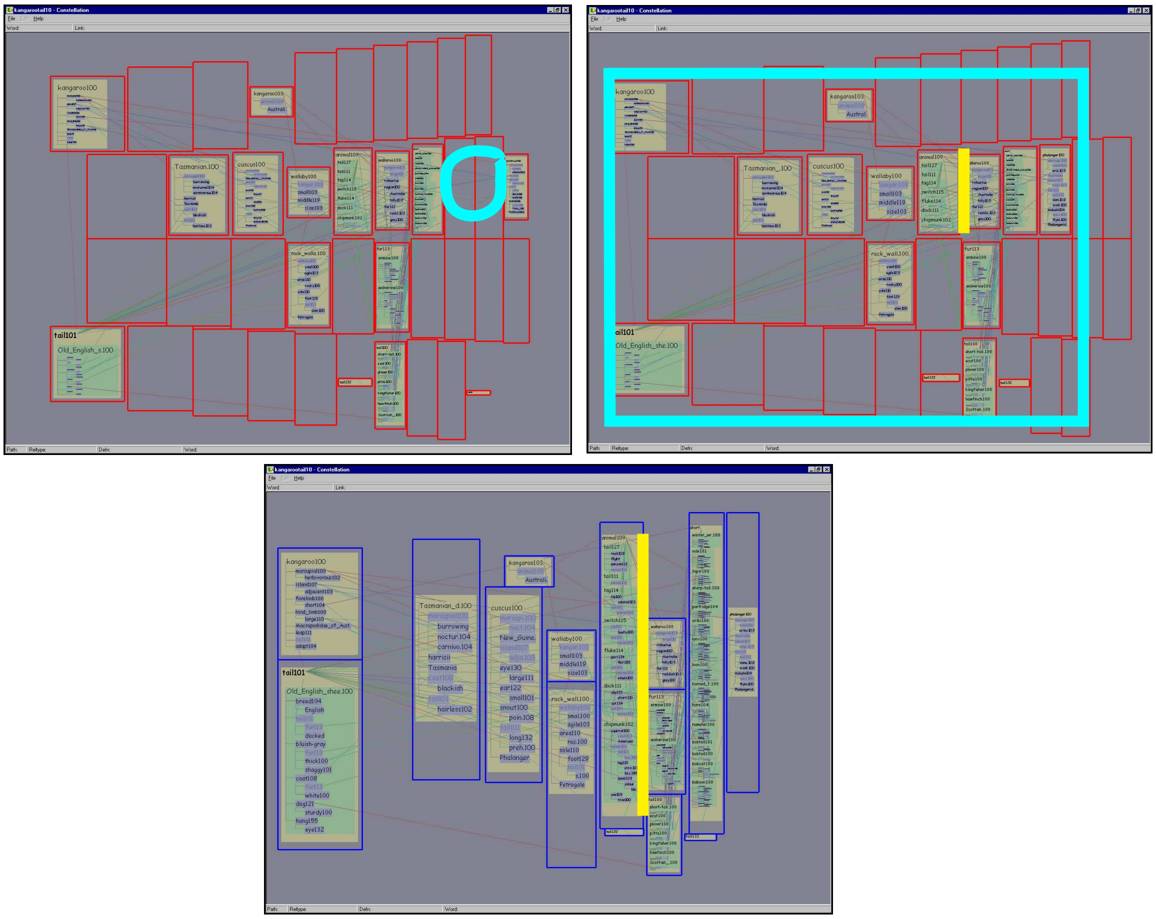

Figure 5.13: **Adjusting grid for maximum information density. Top Left:** This view is already an improvement over the very sparse layout in Figure 5.12, because the singleton path segments have been elided. In the fullscreen view, over 60 words are legible. However, further improvements are still possible. The cyan circle shows a horizontal gap between the 7th and 10th band. **Top Right:** Removing the horizontal gap results in a more compact grid. Here we maintain the identical window borders in all three screen shots to facilitate size comparisons, but the cyan outline shows the borders that would normally be used. 80 words are now visible in the fullscreen view. **Bottom:** All cells have been vertically expanded that contained words of less than maximum size and were also bounded above and below by unoccupied neighboring cells. The vertical yellow lines in this and the previous shot mark a cell that has grown much taller, allowing the words to be drawn more legibly. This fullscreen view contains over 170 legible words.

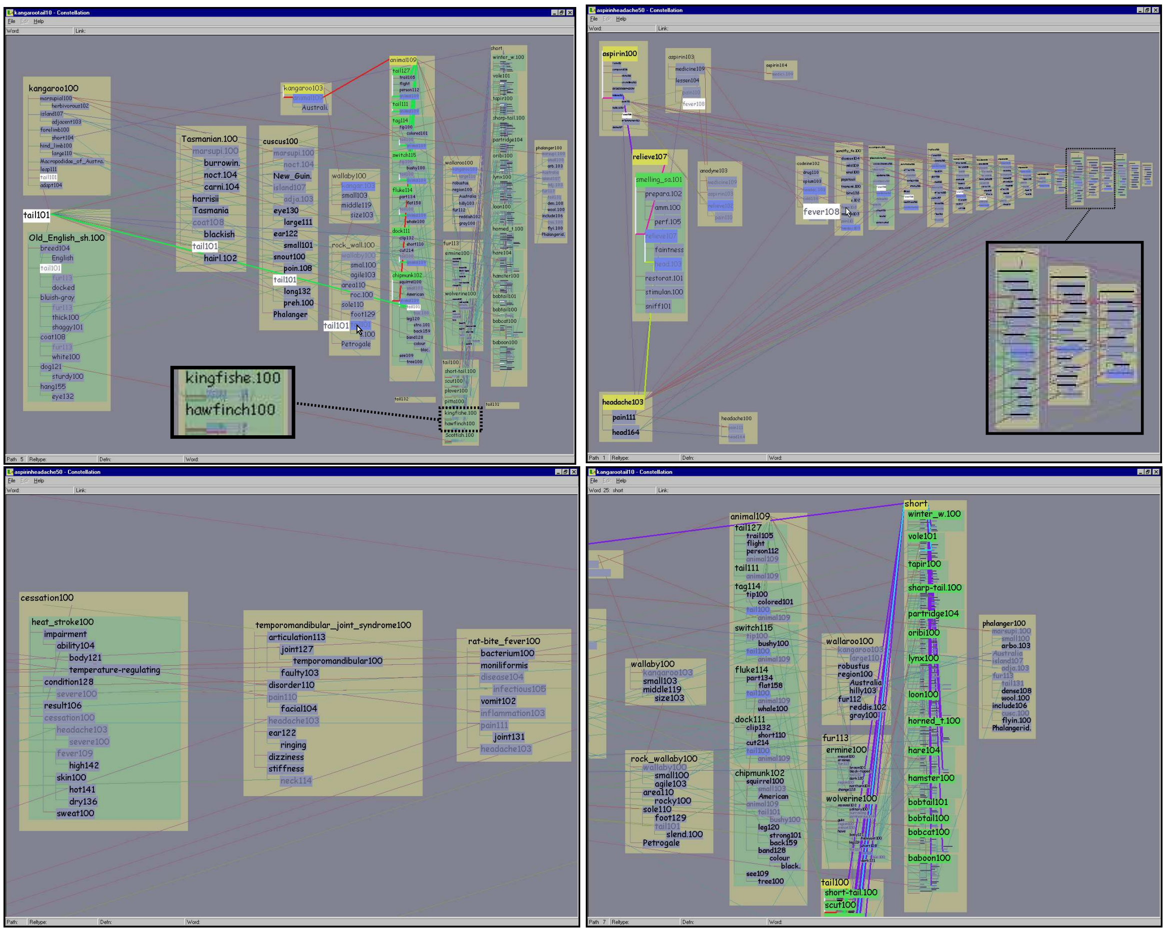

Figure 5.14: **Viewing levels. Top Left:** The overview level is optimized for showing global path structure, and the inset shows that in this case leafword text may be omitted completely. In this 10-path KANGAROO TAIL dataset all instances of TAIL101 are marked by the hovering cursor. The highlighted path 5 constellation shows that many green definition graph boxes are associated with the pathword ANIMAL109. **Top Right:** We avoid sudden jumps in visual salience with a greeking step between the complete omission of leafword text and the smallest size text font. In this 10-path ASPIRIN and HEADACHE dataset, the first path constellation is highlighted, and all instances of the word FEVER108 are marked by the hovering cursor. **Bottom left:** The definition graph reading level is shown for the same area as the inset above, and the different aspect ratio allows every word in the tan boxes to be easily readable. **Bottom right:** The intermediate path segment viewing level is shown with path 7 highlighted, emphasizing the legibility of the definition graph headwords at the top of each green box.

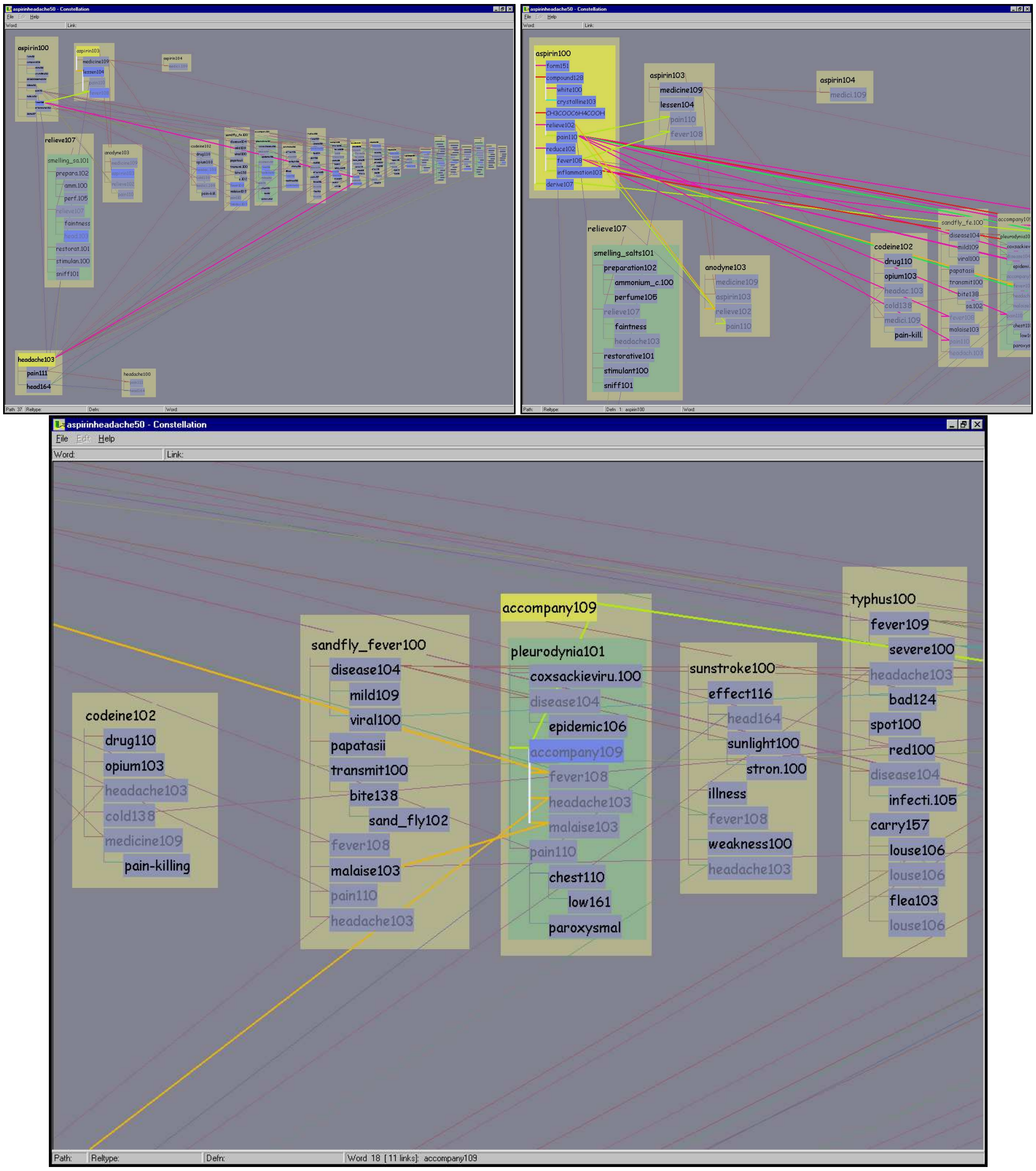

Figure 5.15: **Constellations. Top Left:** Path 37 of the ASPIRIN-HEADACHE 50 path dataset is highlighted. **Top Right:** The definition graph for ASPIRIN100 is highlighted here. Every long-distance link between a master word in that constellation and all its proxies is also highlighted to underscore the relationship of the shared words. **Bottom:** The constellation of all words connected to ACCOMPANY109 is highlighted in this partially zoomed-in view.

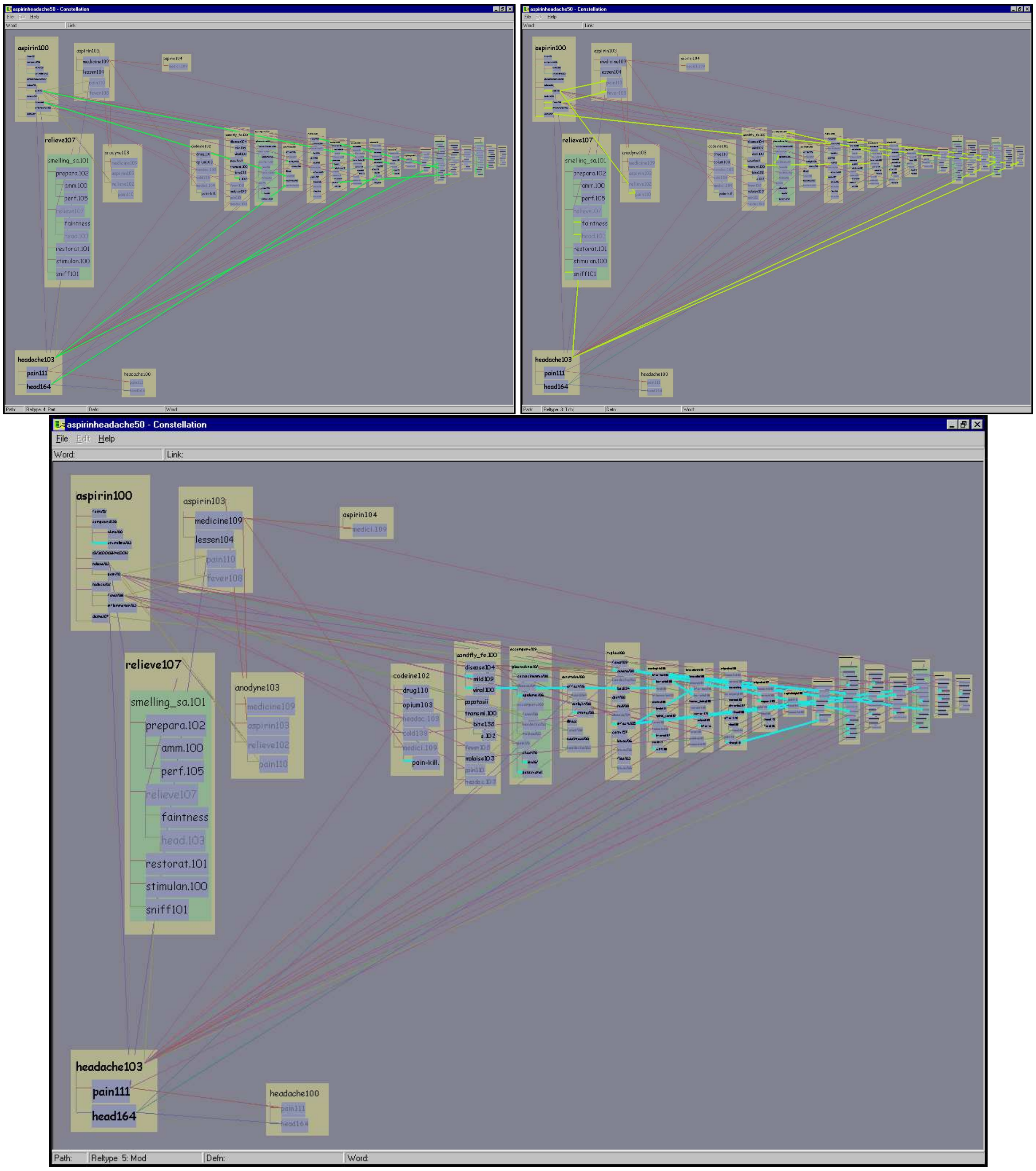

Figure 5.16: **Relation type constellations.** Dataset of the first 50 aspirin-headache paths. **Top Right:** All relations of type `part-of` are highlighted in green. **Top Left:** All relations of type `transitive object` are highlighted in yellow. **Bottom:** All relations of type `modifier` are highlighted in cyan.

Shape	Instance	Color	Hue	(Saturation, Brightness) Dim	(Saturation, Brightness) Bright	(R,G,B) Dim	(R,G,B) Bright
Lines	is-a	red	0			153, 92, 92	229, 25, 25
	subject	orange	45			153, 139, 92	229, 178, 25
	object	yellow	90			139, 153, 92	178, 229, 25
	part-of	green	135	40,60	90,90	92, 153, 107	25, 229, 76
	modifier	cyan	180			92, 153, 153	25, 229, 229
	location	blue	225			92, 107, 153	25, 76, 229
	join	purple	270			122, 92, 153	127, 25, 229
	other	magenta	315			153, 92, 138	229, 25, 178
Boxes	path	tan	60			178, 178, 143	214, 217, 87
	definition	green	125	20,70	50,90	143, 178, 145	87, 217, 92
	leaf	blue	235			143, 147, 178	115, 134, 229
Other	background	grey	240			129, 129, 143	, ,
	text	grey	166	48,122	0,0	100, 103, 123	0, 0, 0
	pie flipper	grey	240	10,70	10,80	161, 161, 179	179, 179, 199

Table 5.1: **Color scheme used for the visualization, in both HSB and RGB.** Each relation type is coded with hues 45 degrees apart on the HSB color wheel, and the hues for word types were empirically chosen to complement them. The highlight colors are obtained by increasing both the saturation and brightness. Hues range from 0 to 360, saturation and brightness range from 0 to 100, and red/green/blue values range from 0 to 255.

brightness level than their background labels so that they are inconspicuous from afar.

- **Hue Discriminability:** The hue of each item should be as distinct as possible. Hues are more discriminable in large than small areas.
- **Text Legibility:** Legibility is increased by a strong contrast in both saturation and brightness between the text and its background. We always render black text in a color coded box to increase both legibility and discriminability.

5.4 Interaction

The two main interaction techniques in Constellation are navigation and interactive visual emphasis through selective highlighting.

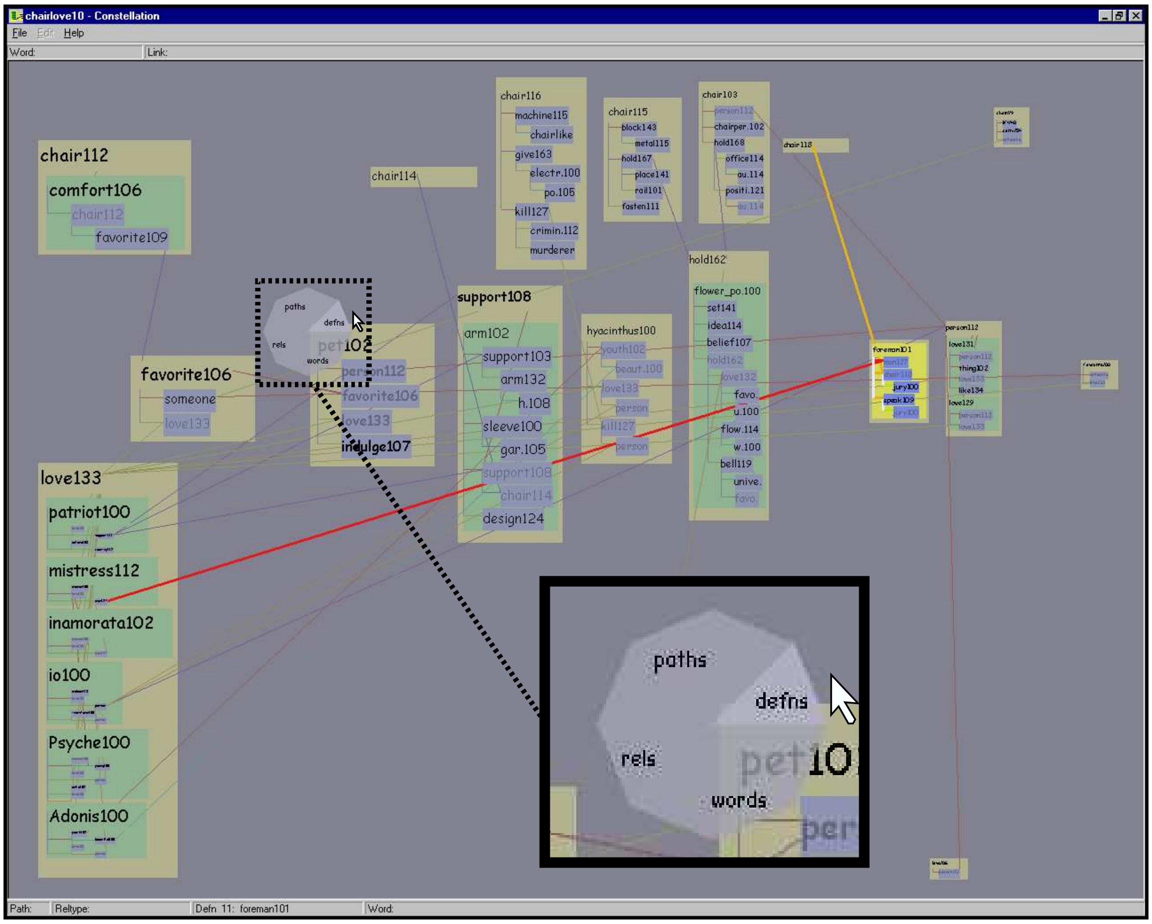

Figure 5.17: **Pie flipper.** The inset shows a translucent radial popup display for flipping between the instances of constellations in a category. The FOREMAN101 definition graph constellation is being chosen in the CHAIR-LOVE 10 path dataset.

5.4.1.2 Hovering

Our extremely lightweight hover mode allows quick visual inspection with no need to navigate. In hover mode, simply moving the mouse over a link marks it visually and shows full details about its origin and destination in an upper status bar. This functionality, shown in Figure 5.18, was added after a direct request from the linguists, who wanted to see offscreen information without needing to navigate there and back when zoomed in to read a definition graph. Hovering over a word will temporarily draw it at maximum size so that it is legible even from the overview position, and visually mark all proxy versions of the word by drawing their label backgrounds as white instead of blue. Again, the linguists requested this functionality to aid reading while avoiding navigation. They wanted to study shared word relationships from the overview

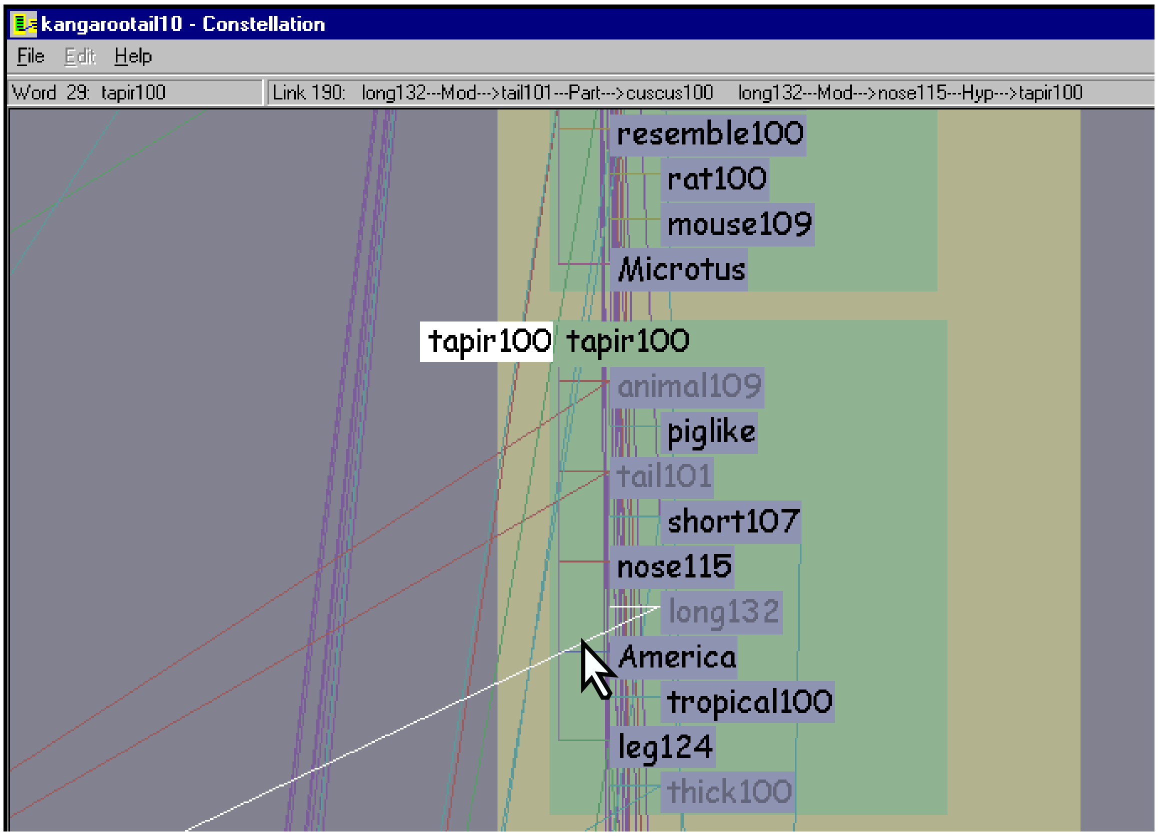

Figure 5.18: **Hovering.** Holding down the shift key enters the lightweight hover mode, where simply moving the mouse both visually marks it in white and yields more information about the object beneath it. Moving the mouse over a link marks it and shows detailed information about its origin and destination in an upper status bar. The mouse is also inside the green definition box for TAPIR, so it is marked by being drawn large on the left. If there were multiple instances of the word, all would be marked, as in Figures 5.14.

position without zooming in to read small words. The large word is drawn to the left of the box so that the large word does not occlude the words directly below the highlighted one, as in Figure 5.14.

5.4.2 Navigation

The main navigation method is a mouse click inside any enclosure box, which triggers an animated transition. Such transitions are important for helping the user maintain mental context [RCM93], which is especially important in our system since zooms usually entail nonrigid motion. The horizontal and vertical zoom scales are computed separately, so that the enclosing box is vertically framed within the window and there is enough horizontal space to to draw every character in all the enclosed labels without elision. Thus a simple click in a definition graph box guarantees that every word in an entire definition is easily readable. The differing aspect

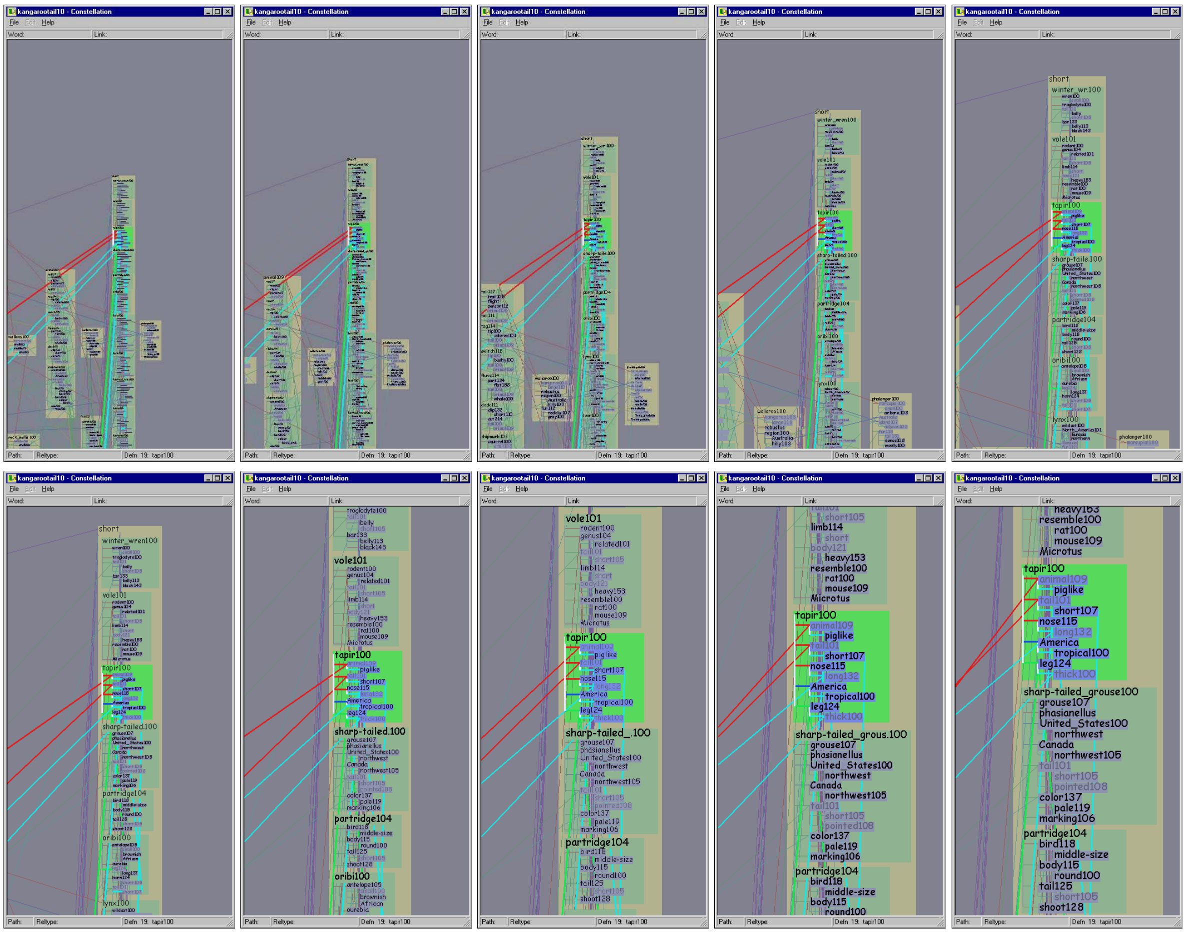

Figure 5.19: **Zooming.** This zoom sequence centered on TAPIR100 shows the adaptive layout in action.

ratios of equivalent regions of the screen depending on the zoom level is visible by comparing the global view at the top left of Figure 5.14 to the view after an animated transition shown below it.

A click in a path segment box will guarantee that every headword is in the field of view, and in most cases that at least every headword is large enough to read. The bottom right of Figure 5.14 illustrates the resulting view.

Mouse dragging offers the user direct control over panning and zooming. The pan control is a left mouse drag. Zooming is either continuous through a right mouse drag when the control key is held down, or in discrete increments using the mouse scrollwheel detents. Figure 5.19 illustrates the gradual change in relative word sizes during a long zoom sequence.

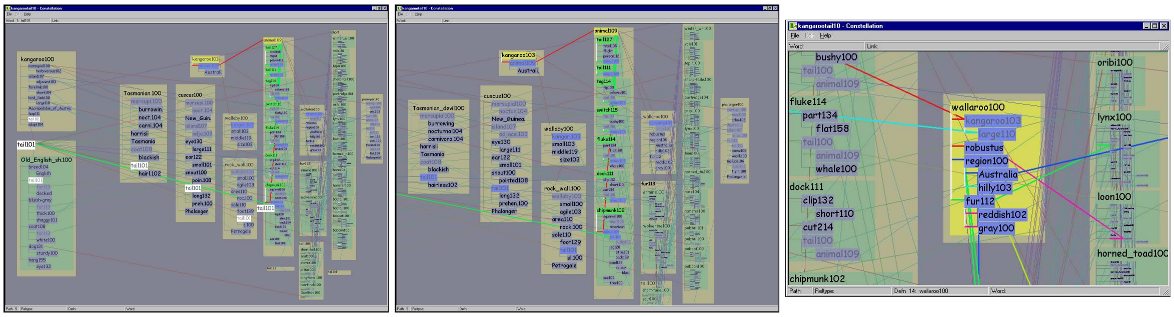

Figure 5.20: **Three effective viewing levels. Top:** The global overview is effective for inter-path comparison. **Middle:** The intermediate view shows the definition graphs associated with a path word. **Bottom:** The closeup view allows the linguists to read an individual definition graph.

To compensate for our finite resolution, we offer easy navigation with animated transitions and intelligent zooming, where the relative amount of space devoted to words changes based on the zoom level. Rapid visual emphasis through hovering is useful in situations where navigation would be a cognitive burden.

The layout provides a great deal of structural information about the paths and definitions that were returned by a MindNet query, at the expense of many edge crossings. Our visual layering approach of using many perceptual channels in concert proved to be quite effective at both avoiding false edge attachments and visual emphasis. The psychophysical literature on color coding is extensive [War00, Chapter 4] [RT96], and we benefited from it by following recommendations of Reynolds [Rey94].

5.6.2 Layout Efficacy

Figure 5.20 shows that we succeeded in creating a layout that was effective at three different viewing levels that corresponded to the three targeted subtasks: a global overview for inter-path comparison, an intermediate view to inspect the definition graphs associated with a particular pathword, and a closeup view well-suited for reading individual definition graphs. Reading the definition graphs, which correspond to a dictionary entry, is a critical part of the plausibility-checking task.

Our spatial layout provides insight into the similarity between the words of the initial query from the global overview level. Figure 5.22 shows the 10 path BIRD-FEATHER dataset, which has the typical sideways "T" shape formed by strongly associated words. The words REGAIN and BANK are quite dissimilar, so there are not many long-distance proxy links in Figure 5.21.

The efficacy of the Constellation views is clear when compared to other pre-existing views of the same dataset. We found that relying on generic graph layout techniques to display this complex structure led to inadequate results. Figure 5.23 left shows part of the kangaroo-tail dataset laid out using *dot* [GKNV93], one

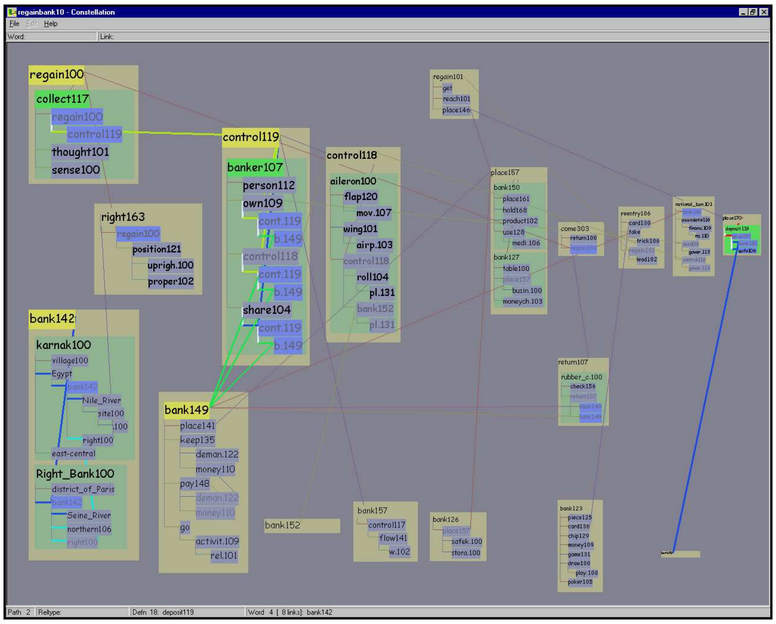

Figure 5.21: **Dissimilar query words.** The query words REGAIN and BANK are dissimilar, so there are few shared words. Three constellations are composed: path 2, a green definition graph for DEPOSIT119 on the far right, and words connected to BANK132 on the far left.

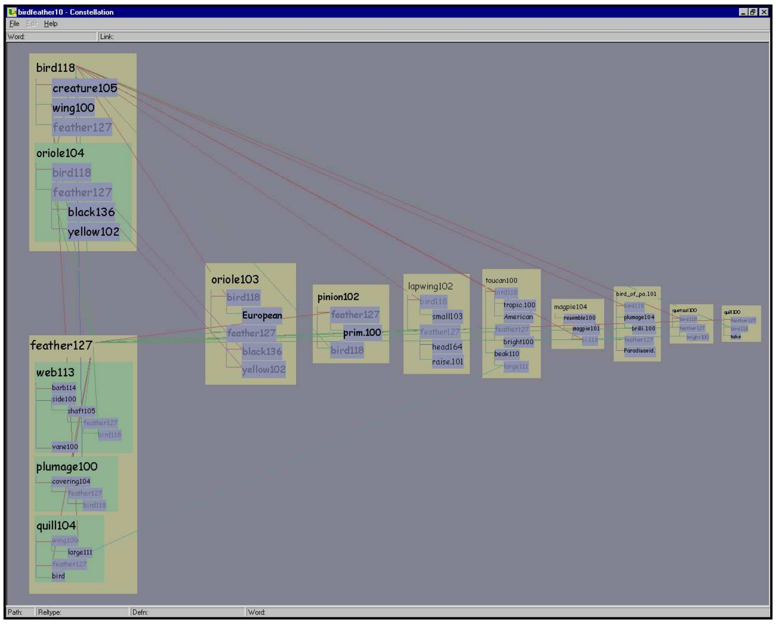

Figure 5.22: BIRD-FEATHER **10 path dataset.** The strongly associated words BIRD and FEATHER results in a long characteristic T shape using our layout algorithm.

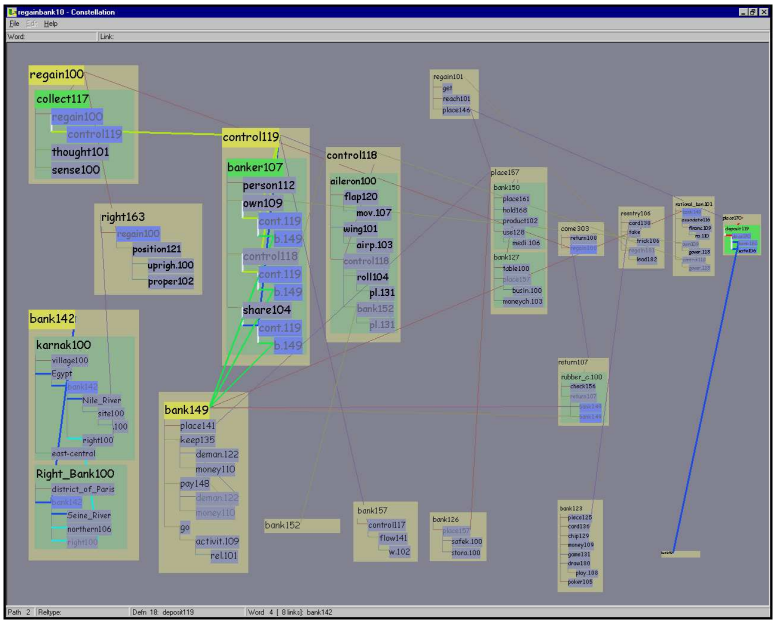

Figure 5.21: **Dissimilar query words.** The query words REGAIN and BANK are dissimilar, so there are few shared words. Three constellations are composed: path 2, a green definition graph for DEPOSIT119 on the far right, and words connected to BANK132 on the far left.

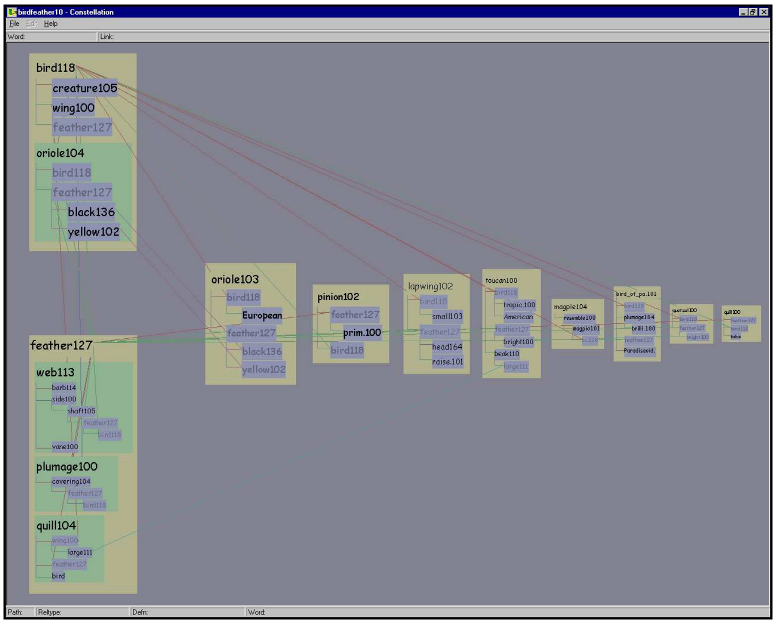

Figure 5.22: BIRD-FEATHER **10 path dataset.** The strongly associated words BIRD and FEATHER results in a long characteristic T shape using our layout algorithm.

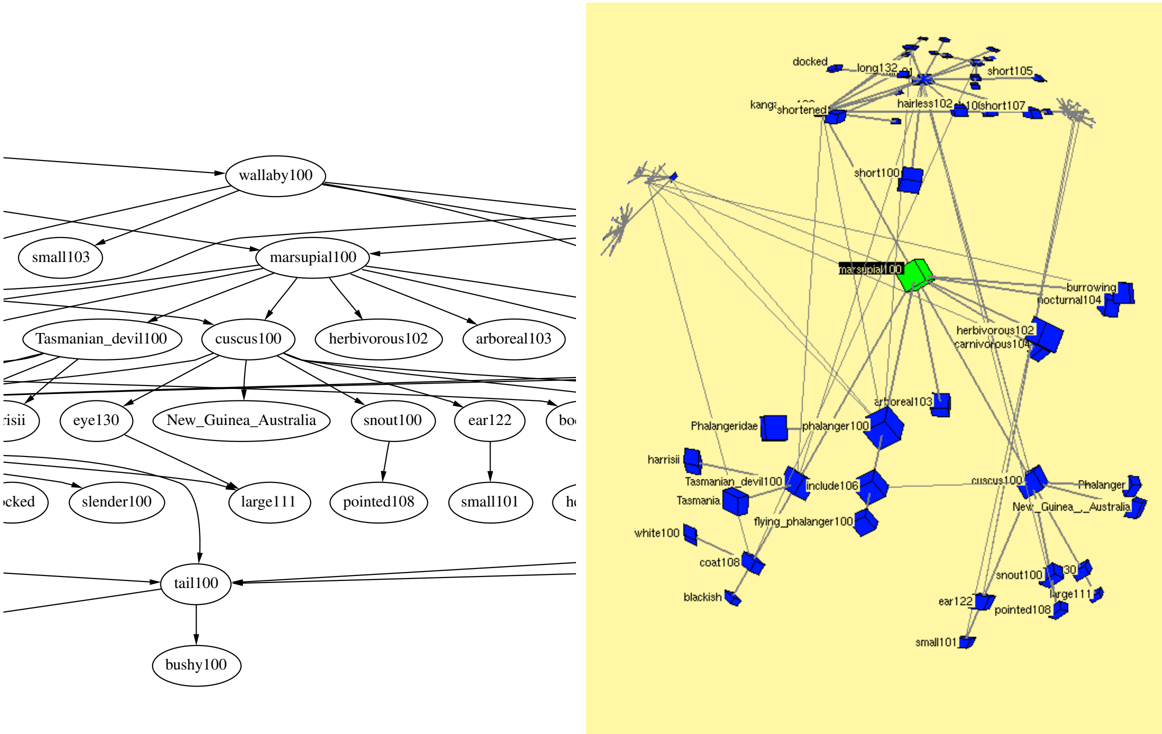

Figure 5.23: **Layouts of kangaroo-tail dataset using pre-existing systems. Left:** Layout using *dot*, one of the more flexible and scalable 2D graph layout systems. **Right:** Layout using our H3 system. Neither layout is effective for a linguist making plausibility judgements about paths or reading individual definition graphs.

of the more flexible and scalable 2D graph drawing systems. Figure 5.23 right shows the same dataset laid out in H3. Both views show an aspect of the graph structure, but neither are suitable for making plausibility judgements about paths by reading individual definition graphs.

5.6.3 Outcomes

Constellation is not in active use by our small target group of linguists, whose project goals shifted during the time that we built the visualization system.

Our target group was quite available during the first phase of the project when I was a summer intern at Microsoft Research in 1998. My coauthor and I were able to obtain more feedback on later prototypes during several visits to MSR over the course of the next year. Unfortunately, by the summer of 1999, the project goals of the linguists had shifted as new aspects of their research came to the fore, and plausibility checking was no longer a major task.

Although Constellation was designed for a small target audience, our design principles are relevant for

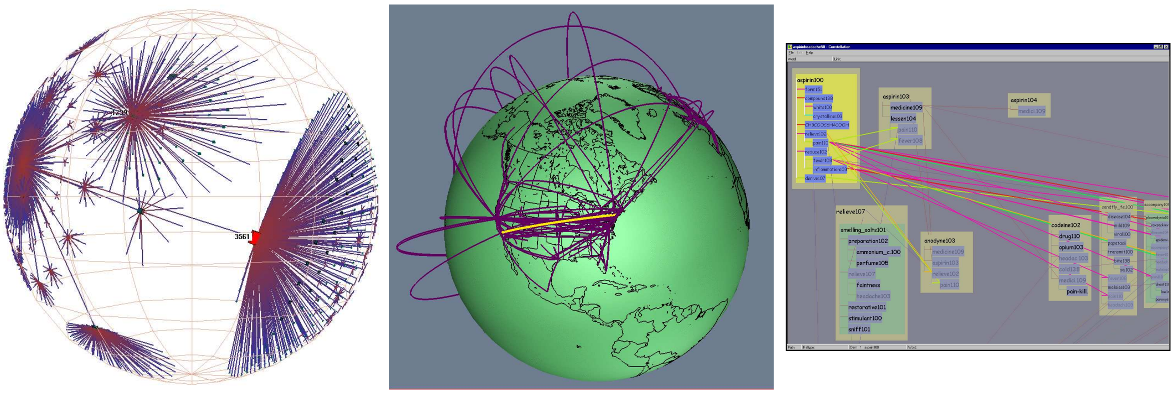

Figure 6.1: **Left:** In H3 the aggregate blobs on the fringe pop out, showing distant points of possible interest. **Middle:** In Planet Multicast the long distance tunnels are the most visually salient. **Right:** In Constellation the foreground layer pops out from the background layer because of the deliberate design decision to use multiple perceptual channels in concert.

Constellation system, we deliberately designed an interaction mechanism that results in a foreground layer popping out from a background layer, which was discussed in detail in Section 5.4.1.

The three preceding cases highlight positive examples of our design, where we successfully encoded the intended domain semantics into visually salient features. We also have to evaluate our visual encoding choices to ensure that we do not inadvertently lead the viewer astray: "... inappropriate perceptual organization can lead to false graphical implicatures if the viewer is led to draw incorrect inferences due to the presence of misleading perceptual groupings or orderings." [Mar91, p. 400]

One of the earlier prototypes of the Constellation system had a visually salient artifact that caused the users to draw the wrong conclusion. Figure 6.2 shows the problem: the word SHORT is much larger than the word FORELIMB100, which was a misleading visual cue that led the linguists to infer that one word was more important than the other. However, these two words have the same computed importance: the size difference was caused by the irrelevant fact that one word had several more letters than the other. The original drawing algorithm made font size decisions on a word by word basis, where we used the largest font that would fit into the allocated screen area for that word. Although this decision had seemed reasonable when we were focused on maximizing information density, after observing the linguists using the prototype we switched to the algorithm discussed in section 5.2.9, where words of the same importance are guaranteed to be drawn the same size. In this more sophisticated multiscale layout algorithm, we achieve high information density by changing the relative word size in response to user navigation, as in section 5.2.10, while avoiding visual artifacts.

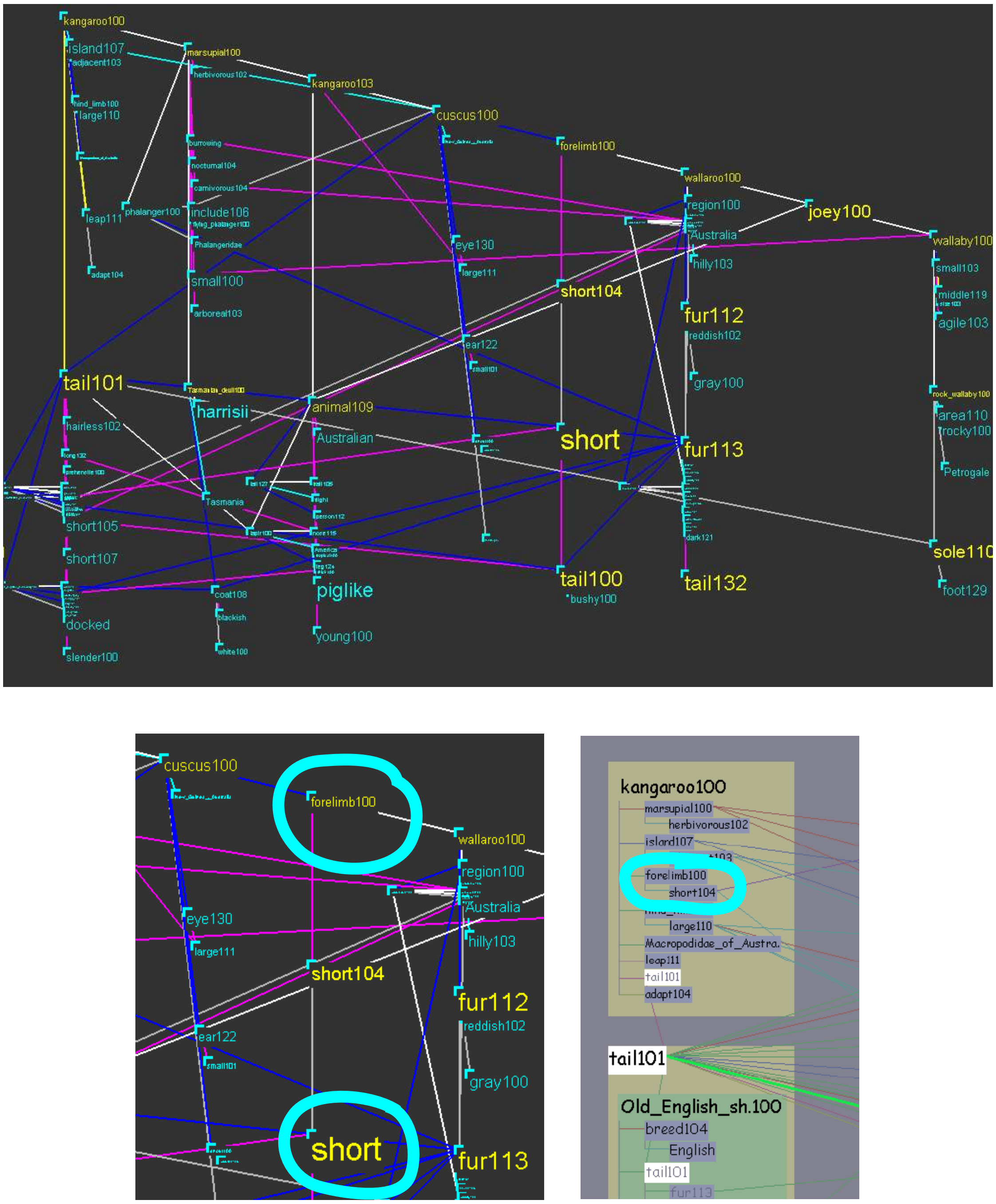

Figure 6.2: **Top:** A very early layout attempt that led the linguists to incorrect conclusions. **Bottom left:** The word SHORT contains fewer letters than the word FORELIMB100, so a larger font fits in the allocated space. **Bottom right:** Words of the same computed importance are always drawn the same size in the final version.