Bead: Explorations in Information Visualization

Matthew Chalmers & Paul Chitson[†]

Rank Xerox Cambridge EuroPARC
61 Regent Street, Cambridge, CB2 1AB, United Kingdom

1 Abstract

We describe work on the visualization of bibliographic data and, to aid in this task, the application of numerical techniques for multidimensional scaling.

Many areas of scientific research involve complex multivariate data. One example of this is Information Retrieval. Document comparisons may be done using a large number of variables. Such conditions do not favour the more well-known methods of visualization and graphical analysis, as it is rarely feasible to map each variable onto one aspect of even a three-dimensional, coloured and textured space.

Bead is a prototype system for the graphically-based exploration of information. In this system, articles in a bibliography are represented by particles in 3-space. By using physically-based modelling techniques to take advantage of fast methods for the approximation of potential fields, we represent the relationships between articles by their relative spatial positions. Inter-particle forces tend to make similar articles move closer to one another and dissimilar ones move apart. The result is a 3D scene which can be used to visualize patterns in the high-D information space.

C.R. Categories: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.5 [Computer Graphics] Computational Geometry and Object Modeling; I.6 [Simulation and Modeling]: Applications.

Additional key words and phrases: visualization, information retrieval, particle systems, N-body problem.

[†]*Current address*: School of Information Systems, University of East Anglia, Norwich, Norfolk, NR4 7TJ, United Kingdom.

2 Background

The work described in this paper draws upon techniques from several areas, including information retrieval, numerical optimization and computational physics. These techniques are combined in order to form a prototype system for the graphically-based exploration of a complex information space. In short, this work explores the viability of the concept that one can use particles in 3-space to represent documents in ‘information space’, and can set up rules of physical behaviour for these particles (e.g. force and motion) which are driven by the documents’ characteristics. These rules are chosen in order to make of spatial proximity approximate thematic similarity. This metaphor is used to interpret word-based information using the everyday phenomena of spatial position and colour. The following subsections describe the basic concepts underlying this work.

2.1 Numerical Techniques for Structuring Data

Most of the work done on the graphical exploration of complex information spaces has been in the realm of Scientific Visualization [McCormick 87]. These techniques have been developed for fields such as physics, engineering, aerodynamics and chemistry. In a manner complementary to more traditional analytic approaches to the treatment of data, visualization gains its power by drawing upon our everyday cognitive skills. The accent is on providing the user with the opportunity to discern interesting features in the analysed information. This is most straightforward when the data to be explored is of low dimensionality and has a regular or well-structured domain. Slightly higher levels of dimensionality — beyond the three used for spatial position — are often tackled by employing features such as colour and texture. More recently, attempts have been made to embed 3-dimensional coordinate systems within others [Feiner 90].

Such images become increasingly complex as further dimensions are employed. Ultimately one must either restrict the number of dimensions considered, or one must map some number of these dimensions onto a lower, more tractable number. The latter approach is the basis of several techniques in Multivariate Analysis including multidimensional scaling (MDS). As described in [Chatfield 80], MDS is “a term used to describe any procedure which starts with the ‘distances’ between a set of points (or individuals or objects) and finds a configuration of the points, preferably in a smaller number of dimensions, usually 2 or 3.” Such a configuration may then afford the graphically-based, exploratory styles of manipulation usually reserved for lower-dimensional data.

Standard techniques for MDS usually involve an eigenvector analysis of an $n \times n$ matrix, where n is the number of points to be configured. This $O(n^3)$ analytic procedure generates the required configuration of points in a single step, but changes or additions to the set of points require re-execution of the entire analysis. As mentioned by Chatfield & Collins with regard to a special case of MDS, ordinal scaling, one can treat the configuration task as an optimization problem. This has allowed the introduction of iterative numerical techniques such as the *steepest descent* method [Press 88].

Steepest descent models a familiar physical situation, where objects respond to the forces upon them by moving down the local gradient towards states of lower energy. As an example, a ball on a slope rolls downhill. The method therefore has an intuitive appeal, especially when the simulation is to be graphically presented to people wishing to maintain some mental model of the dynamics of their data. One problem of steepest descent is that it may be unable to climb out of a local minimum. The points become stuck in a configuration which disallows progress towards a ‘better’ configuration. A rock on a hillside may obstruct a ball rolling down into a valley.

One technique suited to such situations draws upon a process for hardening metals known as *annealing.* Metal can be raised to a high temperature where its consituent atoms can move relatively freely: it melts. By gradually cooling the liquid, the atoms have time to move towards crystal lattice positions of low energy. The thermal mobility of atoms can allow escape from local minima. As the temperature is lowered and mobility decreases, the probability of such escapes decreases. Gradual cooling leads to the configuration of minimum energy i.e. the optimal state.

Applying these principles to numerical optimization led to the Metropolis algorithm for simulated annealing [Metropolis 53]. This requires a way of describing possible configurations (e.g. the spatial position of points), a method of generating random changes in the configuration (e.g. perturbation of position), an objective function, analogous to energy, whose minimization is the goal of the procedure (e.g. the error between the actual and desired inter-point distances), and finally a control parameter analogous to temperature, in tandem with a schedule for its reduction.

As the schedule progresses, random changes in configuration are successively generated. At each step the change in system energy is calculated. If the step lowers the energy then the change is accepted. If the step increases the energy then the change is accepted with a probability which depends on both the size of the energy step and the current temperature of the system. ‘Uphill’ steps are taken less often as temperature decreases.

Simulated annealing has been used in a variety of application areas to good effect [Kirkpatrick 83]. These areas include VLSI layout, the Travelling Salesman problem, and, of course, statistical mechanics. More recently it has been used in clustering tuples in databases [Bell 90]. Bell et al. reported on techniques for moving tuples of data (each representing a document) between files in order to bring similar tuples together and so enhance responsiveness to queries. They found that annealing produced good results but was computationally expensive.

2.2 Structuring Textual Information

In the vector space model [Salton 89], each document D_i is described by a vector of t terms $\{a_{i1}, a_{i2}, ..., a_{it}\}$. Each term usually corresponds to a unique word or word stem in the document, and has associated with it a coefficent of its ‘significance’ in the document. For example, a_{ik} may be set to 1 when the term k appears in the document, and to 0 when the term is absent from the vector. Alternatively, one might use each a_{ik} to store the weighted frequency of occurrence of each term k in document D_i. One can choose to compare the similarity of vectors using a variety of *document distance* measures [Raghavan 86].

Relevance searches usually return a ranked list of documents, with little information as to how each is related to each other or how each is placed in the overall structure of the corpus. Information Retrieval systems tend to give a localised view of data, and there is scarcity of tools available to explore relationship of local characteristics to global patterns.

To some degree an exception to this lack of displayed contextual information is given in a paper on the use of singular value decomposition (SVD) to the organisation of documents [Dumais 88]. The authors used SVD to decompose high-dimensional information on documents into a lower-order representation of about 50 to 150 dimensions. The examples in the paper used a very small database whose approximate structure was presented in two dimensions. The associated ease of interpretation was not extended to larger bodies of information: instead the lower dimensionality was used to allow faster database operations. While SVD is very stable, it is a moderately expensive one-step analytical process. Also, it is sensitive to numerical rounding inaccuracies [Press 88].

In partitioning a corpus into clusters of similar documents, one may represent the members of a cluster by a single term vector. Clusters may be organized hierarchically, with clusters of clusters &c. A variety of techniques for hierarchical clustering and their consequent applicability to document search and retrieval are surveyed in [Willett 88]. One can perform document distance calculations on a cluster much as one would on a single document. One can compare clusters to find which is a good candidate for insertion of a new document. Similarly one can use aggregate vectors in order to reduce the range of documents involved in relevance searches. Clustering and partitioning can be used in themselves for browsing [Cutting 92].

Standard clustering techniques have at their disposal a very high number of dimensions within which to represent documents and clusters. Each dimension allows an axis which potentially can be used to differentiate documents (or clusters). The cost of having such precision available is the complexity of searching and sorting. The use of SVD shows a slight change in the trade-off between precision and cost: lower dimensionality means rougher representation of document relationships but cheaper access and manipulation. One can consider, then, choosing to accept progressively more approximate representations of a corpus as one decreases the dimensionality of the space used. The benefits will be in ease of access and an increasing approach towards the levels of dimensionality we are familiar with in the everyday physical world.

2.3 Using Forces to Create Structure

Physically-based modelling, for the purposes of animation and simulation, has grown by drawing upon mathematical techniques relevant to the physics of motion and deformation [Witkin 91].

The behaviour of a body of matter can be considered as the aggregate of the behaviours of its constituent particles. Features such as elasticity and fracture can be derived from the many interactions at a finer level of detail. Ultimately one becomes grounded in particle dynamics, at the expense of having to consider the many interactions of the particles making up a scene. Potentially, all N particles can lead to $N(N-1)$ pairwise interactions. This *N-body problem* is particularly well-known in computational physics where interactions due to gravitational or Coulombic fields are often of exactly this $O(N^2)$ complexity.

Approximate solutions of lower orders of complexity have therefore been subjects of research activity in recent years. By using hierarchical structures for spatial subdivision such as k-d trees [Appel 85] and octrees [Barnes 86] relatively straightforward algorithms of $O(N \log N)$ complexity have been developed. A slightly more complex algorithm of linear complexity also now exists [Greengard 88]. A region of partitioned 3D space is commonly called a *voxel*, from ‘volume cell’.

All these methods employ the principle of superposition whereby the potential field due to a cluster of particles within a given region of space can be approximated by a single ‘metaparticle’ which is the aggregate of the cluster. The modelled space is subdivided into a tree of voxels containing clusters of particles, each region having its own metaparticle (see Figure 1).

The position of a metaparticle is at the centre of mass of the set of particles inside the voxel. If each particle is defined as having unit mass, then a metaparticles’ mass is the number of particles in its voxel. In calculating the forces acting upon any particular particle, the effects of other particles can be approximated by using progressively larger metaparticles (i.e. higher nodes in the tree) as one considers progressively more distant particles.

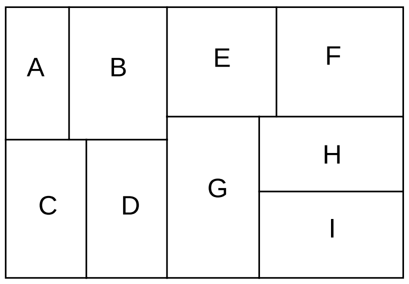

FIGURE 1. A rectangular space is subdivided using a 2-d tree into nine leaf-level subregions. Each leaf is labelled with a single letter. The root node contains all leaf nodes and so might be labelled *ABCDEFGHI.* It is made up of two children, the subregions *ABCD* and *EFGHI.* Similarly *EFGHI* is split into two children *EF* and *GHI, GHI* into *G* and *HI,* and *HI* into *H* and *I.*

A particle in I might then have the forces on it calculated using the other particles in I and the siblings of ancestors in the tree: the metaparticles H, G, EF and $ABCD$. In this manner one can reduce the force computation process for each particle to $O(m + \log(N/m))$ where m is the average size of a leaf and $\log(N/m)$ is the average depth of the tree containing N particles. Hence the overall algorithm is $O(N \log N)$.

Given that we can calculate the force F acting on a particle, then the next position of the particle can be found by solving the familiar second order ordinary differential equation (O.D.E). A variety of numerical solution techniques can be used, such as Euler and Runge-Kutta [Press 88]. With these methods, an O.D.E. is solved for a given time interval. Normally a number of smaller time steps will be taken in the solution process for the required time interval. During the computation the size of the time step will change in attempts to control the accuracy of the solution.

Such algorithms make more tractable simulations involving potential fields and forces. They aid the modelling of complex dynamics where, from little *a priori* positional information, the forces between particles lead to the emergence of structures and patterns in the modelled data.

3 Combining Concepts

The generic nature of the discretization at the basis of fast N-body techniques can be applied to Information Retrieval. We associate documents with particles in space, and use two concepts to underpin this work: a cluster of such hybrid objects can be represented by a metaparticle, and when comparing particles, the difference between the actual ‘geometric’ distance and the desired ‘document’ distance can be used as the basis for a potential field.

We use a model of a damped spring in order to generate forces of attraction and repulsion between particles. When particles are too close the spring pushes them apart, and when they are too far apart they are drawn towards each other. In each case the spring works to regain its ‘rest distance’ which we use as a metaphor for the document distance. We use hierarchical spatial subdivision in order to speed up the calculation of the interactions between the full set of particles. We also use the hierarchy to form document clusters, using addition and normailization of term vectors.

The raw data we have been using is drawn from the HCI Bibliography Project [Perlman 91]. This project has as its goal the construction of a publicly accessible extended bibliography on Human-Computer Interaction. Rather than perform full-text comparisons at this early stage, we have chosen to use keyword lists. Each keyword in an article α is then stored as an element in a vector $K = \{k_{\alpha 1}, k_{\alpha 2}, ..., k_{\alpha t}\}$ along with a vector of coefficents $C = \{c_{\alpha 1}, c_{\alpha 2}, ..., c_{\alpha t}\}$ such that $|C| = 1$. At present we weight keywords equally i.e. for all m,n, $c_{\alpha m}=c_{\alpha n}$. The version of the database we obtained has 301 articles with such keyword lists.

We denote the set of documents being modelled in the system by the set $D = \{\alpha, \beta, \chi, ...\}$, and the corresponding set of representative particles by the set $P = \{ a, b, c, ...\}$. The document distance function, $\delta(\alpha,\beta)$, that we currently use combines the coefficients associated with pairs of matching keywords $k_{\alpha i}$ and $k_{\beta j}$:

$$\delta(\alpha, \beta) = 1 - \left(\sum_{i,j} (c_{\alpha i} \cdot c_{\beta j}) \right)^2, \text{ such that } k_{\alpha i} = k_{\beta j}$$

After initial experiments with no power term (i.e. using the scalar product) we changed to using the formulation above. The squaring tended to be more revealing of structure. We use Euclidean distance for the inter-particle distance function, $d(a,b)$. We define the spring force magnitude between particles (α,a) and (β,b) as being linearly proportional to $\delta(\alpha,\beta) - d(a,b)$, although for stability we add viscous damping proportional to the velocity of the particle. Using spring constant κ_s, damping constant κ_d and $l = a-b$ (and hence $d(a,b) = |l|$) the force $F_{\alpha\beta}$ on (α,a) due to (β,b), is then:

$$F_{\alpha\beta} = -\left[\kappa_s (|l| - \delta(\alpha, \beta)) + \kappa_d \frac{\dot{l} \cdot l}{|l|} \right] \frac{l}{|l|}$$

The overall force F_α on a particle α is the sum of the forces arising from all such pairwise interactions with the other particles in D. Note that $F_{\beta\alpha} = -F_{\alpha\beta}$.

We use a 3-D tree to hierarchically subdivide the space into which particles are placed. We maintain a metaparticle for each node in the tree, consisting of a particle position, a mass, and a term vector. We use document distances to metaparticles to recursively descend the tree in an effort to insert a new document near to similar documents. We maintain a maximum number of particles inside each leaf node — usually 10 — and split a leaf when that limit is reached. In splitting voxels we try to find an axis and coordinate with which to split the number of particles as evenly as possible.

Finally, we iteratively use fourth-order adaptive Runge-Kutta in applying either a simulated annealing step or a steepest descent step to the particle system. In this way we attempt to minimize the unbalanced force on each particle.In this way we try to find positions where the often-conflicting demands for proximity reach the best available compromise.

While looking at the maximum unbalanced force on any particle in the system is one useful measure of progress, we have also found it informative to look at the residual sum of squares. Analogous to the mechanical stress of the spring system and denoted here as S, this metric is built up by examining each pair of particles α and β. We compare geometric distance $g_{\alpha\beta}$ and document distance $d_{\alpha\beta}$:

$$S = \frac{\sum_{\alpha < \beta} (g_{\alpha\beta} - d_{\alpha\beta})^2}{\sum_{\alpha < \beta} g_{\alpha\beta}^2}$$

The ‘ideal’ value for *S* is zero. This is difficult to achieve, as it is unlikely that the low dimensionality of the modelled space will afford a ‘perfect’ configuration. We assume that an ‘acceptable’ configuration has been reached when continued iteration produces roughly constant levels of stress and maximum force. One must bear in mind, however, that there is always some chance that the gradual motion of particles may slip and slide the configuration over the edge of some local minimum or plateau of energy. New additions to the set of particles as well as the stochastic nature of the cooling process may lead to such an event. Note also that like any such stochastically-produced result, the configuration achieved may vary if different orders of addition or different starting positions are used. Lastly, there is always some chance of there being more than one stable configuration, but damping can be used to avoid infinite oscillation between such states.

4 Lessons from Early Experiments

In early trials of the particle engine, we noticed a problem with the accuracy of our N-body approximation. This evinced itself in an overall drifting of the cluster of particles i.e. even though only internal forces were being modelled, the centre of mass of the system gradually moved.

A basic assumption of the family of N-body approximations is that the farther away in space one goes from a particle, the larger the metaparticle can be used when performing force calculations. In our tree usage, however, there was occasionally a disparity between geometric distances of particles and the length of the path between them over the tree structure. Referring back to Figure 1, a particle in leaf B may be geometrically very close to a particle in E, but is relatively far from it in the tree.

In order to address this we changed to an algorithm which works outward, in spatial terms, from each leaf node and takes into account the geometric distance to each neighbouring node before accepting a metaparticle as a good approximation.

Another concern arising from early experiments related to the way that a large amount of work was being directed towards making all pairs of unrelated particles be a precise distance apart. Enforcing this precision seemed to be a less significant or informative constraint, and therefore we changed the rule invoking force calculations. Now, unrelated particles only interact with each other when their geometric distance is less than the document distance signifying unrelatedness. It is possible that this rule can lead to oscillation, but we decided that the likelihood was low enough and the benefits large enough to warrant the asymmetric rule.

One of the most significant issues in aiding performance is the choice of the schedule for reducing the modelled ‘temperature’ of the system, and we are starting to experiment with various schedules and also a simple facility for manually ‘kicking’ the temperature up by a multiplicative factor.

Another issue is how one adds particles to the simulation. Setting off with all *N* particles randomly positioned within the root voxel is inefficient and unstable: instead one gains an advantage by using the ongoing structuring of the space.We start with just a handful of particles, and alternate between performing one or more steps of the particle engine and adding in new particles. As the structure of the particle system develops we are better able to use the tree of metaparticles to place new particles near to ‘relevant’ neighbours. Currently we take in at most $\sqrt{N}$ new particles at each iteration, and we are experimenting with performing multiple iterations between such additions.

The present system brings down the stress *S* of 301 particles to approximately 0.3. With 100 particles this value is approximately halved. On a Sun SPARCstation 2, an annealing step takes around 18 minutes for 100 particles, 90 minutes for 200, and 150 minutes for 301. While this growth is well below $O(N^2)$, these times are still high.

Document distances are recalculated at every step, and so the most immediate and fruitful adjustment is likely to be the construction of a table of particle-particle document distances. The initial cost of calculation would be amortized over subsequent iterations. Space might become an issue as the size of the database grew, but the present experimental system could comfortably store all the particle-particle distances. Metaparticle distances might also be woven in to such a scheme, but their variability might induce excessive costs.

In the longer term, full-text comparisons will be used, and it would seem necessary to move towards experimentation with a larger-scale corpus and more sophisticated text analyses. Rather than build the requisite tools locally, it is more likely that integration with some of the tools of Xerox PARC’s Natural Language Theory and Technology group will occur [Cutting 91].

5 System Architecture

Bead is made up of a number of processes, all of which execute in parallel and communicate by means of the facilities provided by the Isis Distributed Programming Toolkit [Birman 87]. Figure 2, below, sketches out the system.

Bead makes use of the process group model whereby a process can join (or create) one or more named groups and so receive all messages which are broadcast to them. For example, since the *db* process is in the *iiNewDocs* group, *db* receives information on new documents broadcast (or, more accurately, multicast) to the group. The sender, *newDocs*, does not have to know where on the computer network the members of *iiNewDocs* are. Such location independence makes it easier to take advantage of spare CPU cycles and graphics screens on the local network.

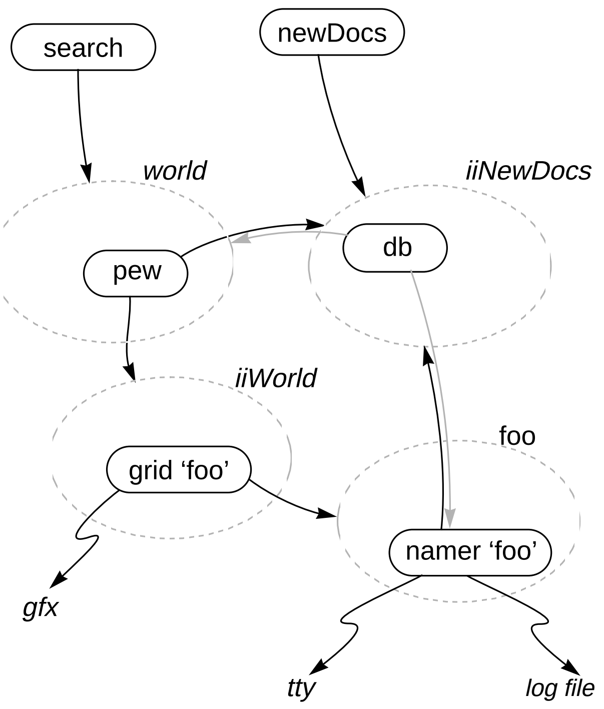

FIGURE 2. Processes (e.g. *newDocs*) communicate with each other by multicasting to named process groups (e.g. *iiNewDocs*). Often a process will obtain information for its own group by sending a request for a multicast e.g. a *pew* obtains new articles to work on by sending a message (shown as a black arrow) to *iiNewDocs*. A member of that group acts upon the message by multicasting the required information to all members of *world* (grey arrow). Computed particle positions are sent by *pew* to *iiWorld*, where each *grid* displays a 3D view. A user can zoom in on a chosen subset of particles, whose IDs can be passed on to a *namer* which outputs more verbose information to a terminal.

The essential numerical calculations of the 'particle engine' are performed by a process called *pew*. It obtains information on articles by communicating with the *db* process (or processes). By joining a group and so receiving messages, a *db* declares and makes active an interest in new documents: hence the group name *iiNewDocs*. At any time, a *newDocs* process can be used to read a file and send information to this group.

The particle engine creates the configuration in 3D which represents the modelled corpus. Once the configuration is stable, interactive visualization and exploration of the corpus can begin. This is done by running one or more tools in parallel with the engine, and having them connected so as to build up the required description of browsed documents.

For each particle, the corresponding position and velocity (and also a colour) are sent to the group of processes which has declared an interest in this 'world'. A number of *grid* processes can join this group and each display a view of the scene. By default the grid viewer shows three orthogonal plots (in XY, XZ and YZ) as well as an animated perspective view as seen from a point of view rotating around the centre of mass.

One way of exploring the particle world is to ask the *pew* processes to make displayable the distribution of a chosen set of query words. A *search* process reads in the words and sends the appropriate message. Each particle is then given a colour according to its document distance from the query, and the resulting shading information is sent on to *iiWorld* e.g. Figure 3.

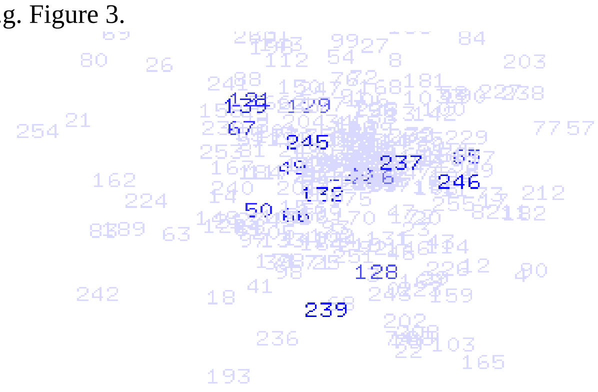

FIGURE 3. A view of the main cluster of documents with a search done on the keywords "information retrieval". From this current point of view one can see a number of matches e.g. article IDs 139, 67, etc. which might be suitable for zooming in on. Note that matching articles are not in a tight cluster but are mixed through the corpus of HCI-related articles.

One can zoom in on a chosen particle in order to see neighbouring particles and how they relate to each other. An adjustable radius defines a 'sphere of interest' centred at the chosen particle. This is used to frame a more confined region of space for display. Also, particles outside the sphere have their colour reduced in intensity slightly so as to allow their patterns and proximity to be noticeable without being imposing e.g. Figure 4.

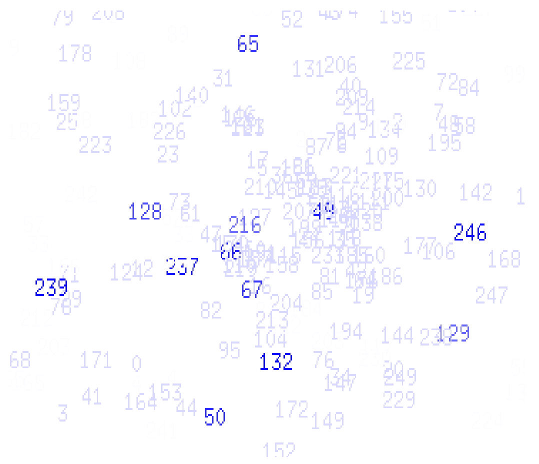

FIGURE 4. Following on from the previous figure we zoom in on 67 (Cone Trees: Animated 3D Visualizations of Hierarchical Information, Card et al., CHI '91). The closest article, 66, is another paper by the same authors.

The remaining 'interesting' particles can also have their IDs sent off to a group (specified individually for each *grid*) where it is expected a *namer* will exist. A *namer* listens for sets of particle IDs, and by communicating with the *iiNewDocs* group obtains information on the original articles e.g. the title, authors, source and date of publication, and the keyword list. These details are printed out to the terminal but can also be logged for later perusal. A command-line version of this program, *whatis*, provides a simple tool for checking on any document id.

The architecture is intended to support varying configurations and extensions to the system e.g. it should be possible to split the workload of the 'particle engine' between more than one *pew* process. Also one can combine different types of processes within a group e.g. a new graphical application with an interest in the particle world might join *iiWorld* without necessitating modifications to *pew* or *grid*. Generally, flexibility of construction makes it easier to integrate and experiment with a greater variety of viewers, filters and loggers independently of the basic data-generating system core.

6 Conclusion

The geometrically-represented document space offers a means for providing users for a 'naturalistic' model of document relationships. The interactive visualization and navigation of the space becomes a means to browse and explore the corpus. This veers slightly away from systems whose aim is more to support retrieval of some small set of relevant items in the corpus which match predetermined characteristics.

We are still building up experience with the present system and making basic improvements and additions. As this work progresses, we will be better able to assess where to concentrate future effort, what should be the focus of usage experiments, what the strengths and weaknesses of the basic modelling process are, and what types of tools and styles of interaction can make best use of such models.

The system in its current state could be used as a component for a prototype 'reactive information environment'. Making the underlying model comprehensible should aid in gaining acceptability for the tools built on top of it. The ultimate goal of this work is to make information spaces explorable both graphically, by humans, and automatically, by programs or 'dæmons'. With these in hand we would go on to investigate how somebody might use a database of document-based information for formal queries as well as more casual, exploratory use. If the ongoing writing and reading of electronic documents can be modelled and shared, then new possibilities for computer-based support of both individual and collaborative work may be opened up.

7 References

[Bell 90] D.A. Bell, F.J. McErlean, P.M. Stewart and S.J. McClean, 'Application of Simulated Annealing to Clustering Tuples in Databases', *Journal of the American Society for Information Science*, 41(2), pp. 98-110, 1990.

[Birman 87] K.P. Birman & T.A. Joseph, 'Exploiting Virtual Synchrony in Distributed Systems', *Operating Systems Review* 22(1), pp. 123-138, 1987.

[Chatfield 80] C. Chatfield & A.J. Collins, *Introduction to Multivariate Analysis*, Chapman & Hall, London, 1980.

[Cutting 91] D.R. Cutting & J.O. Pedersen, 'An Object-oriented Architecture for Text Retrieval', *Proc. RIAO 91 — Intelligent Image and Text Handling*, Universitat Autònoma de Barcelona, Spain, April 1991.

[Cutting 92] D.R. Cutting, D. Karger, J.O. Pedersen & J.W. Tukey, 'Scatter-Gather: A Cluster-Based Approach to Browsing Large Document Collections', to appear in *Proc. SIGIR 92*, Copenhagen, June 1992.

[Dumais 88] S.T. Dumais, G.W. Furnas, T.K. Landauer, S. Deerwester & R. Harshman, 'Using

Latent Semantic Analysis to Improve Access to Textual Information', in *Proceedings of CHI'88*, special issue of *CHI Bulletin*, ACM, pp. 281-285, May 1988.

[Feiner 90] S. Feiner & C. Beshers, 'Visualizing n-Dimensional Virtual Worlds with n-Vision', *Computer Graphics* 24(2), pp. 37-38, March 1990.

[Kirkpatrick 83] S. Kirkpatrick, C.D. Gelatt Jr., & M.P. Vecchi, 'Optimization by Simulated Annealing', *Science* volume 220, no. 4598, pp. 671-680.

[McCormick 90] B.H. McCormick, T. DeFanti & M.D. Brown, Visualization in Scientific Computing, special issue of *Computer Graphics* 21(6), November 1987.

[Metropolis 53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller & E. Teller, *Journal of Chemical Physics* 21, p.1087, 1953

[Perlman 91] G. Perlman, 'The HCI Bibliography Project', *SIGCHI Bulletin* 23(3), pp. 15-20, 1991.

[Press 88] W.H. Press, B.P. Flannery, S.A. Teukolsky & W.T. Vetterling, *Numerical Recipes in C*, Cambridge University Press, Cambridge, U.K., 1988.

[Raghavan 86] V.V. Raghavan & S.K.M. Wong, 'A Critical Analysis of Vector Space Model for Information Retrieval', *Journal of the American Society for Information Science*, 37(5), pp 279-287, 1986.

[Salton 89] G. Salton, *Automatic Text Processing*, Addison-Wesley, Reading, Mass., 1989.

[Willett 88] P. Willett, 'Recent Trends in Hierarchical Document Clustering: A Critical Review', *Information Processing and Management*, 24(5), pp. 577-597, 1988.

[Witkin 91] A. Witkin & M. Kass, 'An Introduction to Physically Based Modelling', *SIGGRAPH'91 Course Notes* 23, ACM SIGGRAPH, 1991.