A Linear Iteration Time Layout Algorithm for Visualising High–Dimensional Data

Matthew Chalmers

UBILAB, Union Bank of Switzerland

ABSTRACT

A technique is presented for the layout of high–dimensional data in a low–dimensional space. This technique builds upon the force–based methods that have been used previously to make visualisations of various types of data such as bibliographies and sets of software modules. The canonical force–based model, related to solutions of the N–body problem, has a computational complexity of O(N^2) per iteration. In this paper is presented a stochastically–based algorithm of linear complexity per iteration which produces good layouts, has low overhead, and is easy to implement. Its performance and accuracy are discussed, in particular with regard to the data to which it is applied. Experience with application to bibliographic and time series data, which may have a dimensionality in the tens of thousands, is described.

CR Descriptors: H.1.2 **[Information Systems]**: Models and Principles - *User/machine systems*; H.5.2 **[Information Systems]**: Information Interfaces and Presentation - *User interfaces*; **[Computer Graphics]**: Methodology and Techniques.

Keywords: layout algorithms, visualization, high–dimensional data, spring models, stochastic algorithms, force–directed placement.

INTRODUCTION

A great deal of information now available to us from our ever–growing networks and databases is complex, multidimensional data. Textual data such as document collections and time series data such as histories of currency fluctuations are growing in number and size. This same growth means that the data is often difficult to access and understand, as traditional data access techniques show little of the patterns within such data: how individual objects are related to each other, whether larger–scale topics or clusters exist, and how data attributes are distributed across such patterns.
Such tasks are at the core of visualisation, and it would therefore seem appropriate to try to apply visualisation techniques. At the core of such an information visualisation task is the generation of a layout in a low–dimensional space which represents the multidimensional data well, affords browsing and overview, and can be produced efficiently. Information design and interaction issues related to making such a layout easy to navigate through and to understand undoubtedly interact with the layout process, but this paper will largely confine itself to the topic of algorithmic efficiency. The focus is on techniques which help to make data analysis as interactive as possible and to make available data sets which are as large as possible.
Background work, on several kinds of data and various techniques for layout and structuring, is discussed in the next section. Following that is a description of a novel algorithm which is efficient and which produces good layouts. Performance of the algorithm and the effects of algorithmic parameters and data characteristics are discussed in an ensuing section. This precedes a brief overview of future work plans and a conclusion.

UBILAB, Union Bank of Switzerland, Bahnhofstrasse 45, 8021 Zürich, Switzerland. EMail: chalmers@ubilab.ubs.ch

BACKGROUND AND RELATED WORK

Information visualisation broke away from its roots in scientific visualisation when researchers began to explore data which did not have a simple set of orthogonal dimensions that could be mapped on to two or three spatial axes plus a few others mapped to such attributes as colour and texture.
The n-Vision system was a move on from using only the basic three spatial dimensions of most scientific visualisation systems [9]. Nested sets of dimensions were presented to the user. At chosen points in a basic three-dimensional space, a further set of three axes would be placed. This nested set of axes could then be used to render a new graph or plot, and also to insert another more deeply nested set of axes. The dimensionality of the financial data rendered in this way was of the order of 10. Note that only the selected points had deeper data rendered as nested dimensions. No overview of the entire data set, using the entire set of dimensions, was possible as the rendered image would be strongly cluttered.
Another step forward from basic scientific visualisation techniques was taken at Xerox PARC and presented in a trio of papers: [1], [14] and [17]. The Information Visualizer paper described a framework for breaking up information based on task activity into different rooms, for controlling and scheduling interaction to offer better interaction to the user, and also gave a lead in laying out data by use of abstract data structures such as hierarchies and common linear dimensions. The Cone Tree and Perspective Wall papers described the implementation and detail of visualisations employing these two data structures. These three papers rightly emphasised the importance of human perception in making visualisations effective. The authors demonstrated static and dynamic features of their information designs which helped with this task. The data structures they used were relatively simple, however, and the shapes of Cone Trees also led to severe occlusion problems.
One of the fundamental works in visualising more complexly structured data is SemNet [8]. Multidimensional scaling (MDS) techniques were used in order to reduce the dimensionality of data on Prolog modules and their inter–calling relationships down to

three dimensions. MDS concentrates on finding a linear combination of the original dimensions which maximises the 'spread' of the data [4]. It blends together dimensions which seem to be strongly correlated. Note that this is a very strong constraint on the layout: ordering according to the final dimensions must be preserved, even if global metrics of similarity, or indirect relationships and clustering, might contradict this ordering. As the authors noted, standard MDS algorithms are slow ($O(N^3)$ or $O(N^4)$), which led them to experiment with a heuristic based on simulated annealing. Also, most MDS algorithms are non–iterative, which means that they have to be re-run in their entirety for adaptive and ongoing applications.

MDS is one of many techniques for dimensional reduction, or layout, which are based on optimisation. An error metric for layout quality is defined, and the algorithm is applied to find a layout with the lowest possible error. The simplest one of these uses a spring model to try to minimise the difference between the geometric distances in the visualised space and distances in the high–dimensional space. Examples are the software modules in [6] and the WWW graphs mentioned in [12]. The springs generate forces in directions which tend to pull distant but similar objects towards each other, and push close but dissimilar ones apart. It is often the case that instead of there being only a few well-defined links from each object to others, every object is connected to some degree to every other one. Therefore the force calculation is $O(N^2)$, which tends to limit the scale of the system modelled.

This is the familiar N-body problem, and gradient descent (also sometimes known as force–directed placement) and simulated annealing are common methods to employ in this situation. For the basics of these techniques, please refer to [16]. Gradient descent is simpler, but occasionally prone to getting stuck in local minima. Annealing is more robust as it tends to 'shake free' of such below–optimal layouts. Techniques such as Greengard–Rokhlin [10] can reduce the force calculation to linear time by using hierarchical spatial subdivision techniques, effectively by discretising the set of objects over 3D space. Maintaining the spatial information can create significant overhead, and integration with annealing is difficult due to annealing's backtracking steps.

An $O(N.logN)$ per iteration annealing algorithm for the layout of textual information, using spatial subdivision, was presented in [2]. Textual data is of high dimensionality, as each word (or term) contained in the set of modelled documents effectively defines an independent axis in a vector space used to describe the set. Each document is represented by the set of words occurring in it and each word's relative frequency of occurrence, collectively known as a term vector. Subject to some normalisation of term vector lengths, the vector space model also allows metrics of high–dimensional distance to be used such as the scalar product [19].

More recently, experiments were run by the author on an algorithm with iteration time linear in N. This uses spatial subdivision in the manner of Greengard–Rokhlin, with gradient descent combined with added stochastic jitter along the lines of [11] to help avoid local minima. In this algorithm, as with that in [2], each modelled object has a term vector to represent a document and also has a particle with a position and velocity in 3-space. A tree of voxels is maintained, with each voxel maintaining a metaparticle (at the centre of mass) and a 'metadocument' (the aggregate of all the documents contained by the voxel).

Although the force calculation for all leaf–level objects can then be approximated in linear time with the use of voxels' metaparticles and metadocuments, the metadocuments are significantly larger than the leaf documents. Since each voxel's metadocument has to represent all the terms contained in all of its leaves, the size of term vectors grows as one goes up towards the root. In experiments, the root voxel's metaparticle would often have a term vector two orders of magnitude longer than that of the average leaf. As

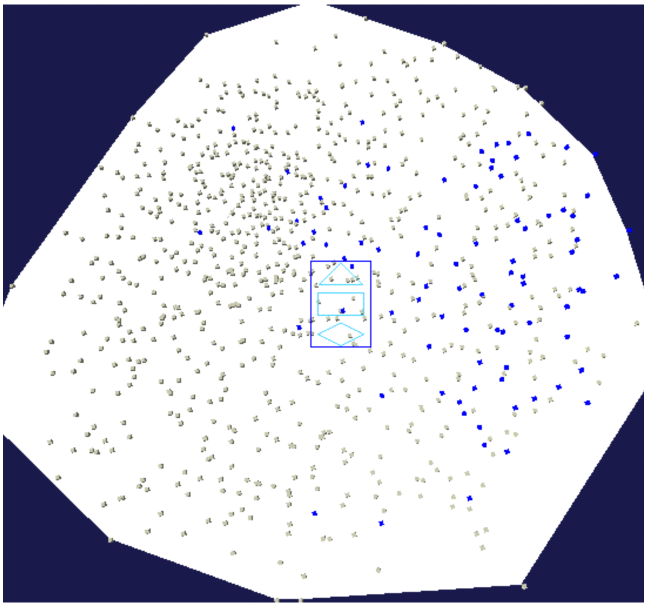

Figure 1. An example layout produced by Bead, seen in overview, of 831 bibliography entries from CHI, CSCW and UIST conferences. The dimensionality (the number of unique words in the set) is 6925 and the layout stress is 0.16. After a search for 'cscw or collaborative' we see the pattern of occurrences coloured dark blue, mostly to the right. The central rectangle is the visualiser's motion control.

objects move, they may change voxel leaf sets, which means updates of the tree have to be done frequently. This also means that the significantly more complex force and high–dimensional distance calculations involving metadocuments cannot be cached. This hampered attempts to scale up to larger data sets.

One possible solution considered was to try to threshold the term vectors so as to only represent more heavily weighted (frequently occurring) words higher up in the tree. This might be combined with a maximum length of term vector for every metadocument. It seemed that many tree maintenance overheads would still be present, however, and experiments with a technique complementary to the discretisation methods of spatial subdivision were promising enough to abandon the voxel-based techniques for the moment. This technique, which avoids many of the overheads associated with discretisation, is based on stochastic sampling. It is described in the following section.

Other techniques have been applied to visualising sets of documents. Kohonen maps were used by [13] in an iterative layout technique which requires at least $O(N^2)$ per iteration and has not been shown to handle larger data sets. More recently, [20] described a technique whereby documents are clustered and the cluster centroids laid out using either principal component analysis or MDS. Individual documents are then scattered or interpolated near to their associated centroids in order to visualise the data set. Computational complexity is much reduced, but the technique tends to ignore the detail of inter–document relationships by relying on the coarse discretisation offered by this initial clustering. Choosing the number and acceptable sizes of clusters to partition the set can be a difficult, data–dependent task, although the techniques of [18] might help in this regard..

STOCHASTIC SAMPLING AND NEIGHBOUR SETS

In this section, we build upon gradient descent to iteratively improve the position of each of N objects involved a layout. Spring-based forces F_{ij} between each pair of objects i and j act to improve interobject separation in the layout. The magnitude of each F_{ij} is linearly proportional to the magnitude of d_{ij} - g_{ij}, where g_{ij} is the current geometric distance between i and j in the layout, and d_{ij} is their high–dimensional distance. An example layout is shown in Figure 1.

In the case of bibliographic data, we use $d_{ij} = (1 - s_{ij}^{p})$, where s_{ij} is the scalar product of the term vectors of i and j, and p is an exponent used to increase the discrimination of the distance metric. Currently we use $p=4$. For time series data we are currently experimenting with a simple metric akin to the scalar product as well as an information theoretic metric, mutual information. The distance metric d_{ij} will be discussed further in a later section.

Instead of doing all the possible $N(N-1)$ pairwise force calculations as in the standard technique, we do force calculations between each object i and the members of two sets whose size is bounded by a constant. In this way, we maintain a computational cost for each iteration which is linear with respect to N

The first set is stored as a dynamically–maintained list of references to 'neighbour' objects, V_i. This list is of maximum length *Vmax*, and entries in V_i are stored in order of distance in high–dimensional space. V_i is initially null. Along with V_i is kept a value, $maxDist_i$, which is the maximum distance to any member of V_i. While the neighbour set is carried over between iterations, the second set is constructed anew each time. This is a randomly chosen subset S_i of all the objects, where *Smax* is a constant and no member of V_i is in S_i.

As each candidate element j for the set S_i is selected, the distance d_{ij} is calculated. If $d_{ij} < maxDist_i$ then j is inserted into the appropriate position in the set V_i instead of S_i, perhaps forcing out the most distant neighbour in the process. Otherwise j is added to S_i, and once we have *Smax* members then the forces on i are calculated using a number of calculations constrained to be less than or equal to the constant *Vmax*+*Smax*:

$$F_i = \sum_{v \in V_i} F_{iv} + \sum_{s \in S_i} F_{is}$$

Extra weights can be applied to shift the balance between the summations over V_i and S_i in the force calculation, or to make this summation more closely approximate the full N-body calculation, but we have not found it necessary to employ them. Typical values used in our system are $Vmax=5$ and $Smax=10$.

Note that as sampling continues, the set S_i evolves towards the set of *Smax* objects most closely related to i. If additions to the data set are made between iterations, they can easily but gradually be accommodated — without performing a global recalculation of complexity greater than linear. If we do not expect further additions to the set, we would generally terminate the layout process after roughly N to $3N$ iterations or terminate it automatically based on having little decrease over several successive stress measurements.

An additional force whose effect is gradually added over the first $2N$ iterations is another spring force between each particle and the plane i.e. a force is added which is proportional to the height above (or depth below) the plane. This force gradually makes the layout shift from being a 3D point cloud to being a near-planar '2.1D' layout, while allowing unassociated clusters to slide away from each other — unlike the overlaying that would happen with projection onto the plane.

Given the summed forces acting on each i, we can perform an integration step to update its position and velocity. Note that sampling adds jitter in a way which helps break out of local minima. For stability we add to each object viscous damping proportional to its velocity, and we also clip excessively high forces. We normally employ 4th order Runge-Kutta integration, although the simpler Euler step is often adequate. Note also that some symmetries aid in

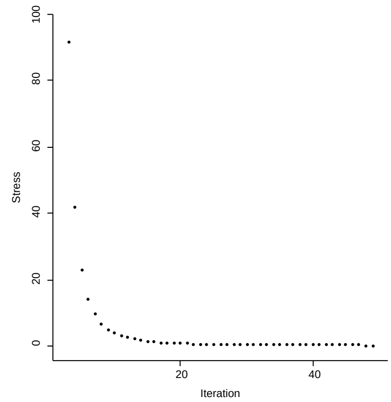

Figure 2. Stress falls quickly against iteration number in the first iterations of the example layout. Before any layout work was done, the stress was 4102.9, but this and two following values (1,1284.4 and 2,274.6) have been elided for legibility. After 56 iterations, the stress was 0.23, and at 896 it was 0.16.

efficiency e.g. $F_{ij} = -F_{ji}$. Also, a new neighbour j for an object i may suggest that i should be evaluated as a potential new neighbour for j. We are looking at work such as [5] to see if epidemic–like algorithms can efficiently spread the news of new arrivals through the network of neighbours.

Objects are then meshed together with a Delaunay tesselation so as to make an island-like landscape. Occasional rough patches can then more easily be seen, where either the layout algorithm has not been able to find a good planar layout or the data's mutual distances do not permit a planar layout. Design issues for such landscapes are the subject of [3]

EVALUATION AND EARLY EXPERIENCE

At the centre of most evaluation of the layout algorithm is a metric based on the mechanical stress of the spring system. It is essentially the residual sum of squared errors of all interobject distances, but has some normalisation which favours more compact layouts:

$$Stress = \frac{\sum_{i<j} (d_{ij} - g_{ij})^2}{\sum_{i<j} g_{ij}^2}$$

The relationship to the stress minimisation involved in the standard spring-based layout algorithm should be clear. Note also that the stress calculation is itself an $O(N^2)$ operation. Consequently we normally only perform it every $\sqrt{N}$ iterations.

In the example layout of 831 bibliography entries from CHI, CSCW and UIST conference proceedings taken from the HCI Bibliography [15] shown in Figure 1, stress levelled out at 0.15. A chart of stress against iteration number is shown in Figure 2. The average iteration time, 6 seconds, is one of the data points shown in the chart of iteration time growth with N in Figure 3, below. Modified to calculate the force on each particle in the standard quadratic way but running on the same data set, the iteration time was 372 seconds.

The layout program, written in C, was run on an SGI High Impact with a MIPS R4400 250MHz processor and R4010 floating point chip. All runs used *Vmax*=5, *Smax*=10, and 4th order Runge-Kutta integration This program can either dump a layout file or position objects within a DIVE virtual environment [7]. Objects in visualised layouts can have their titles shown upon mouse selection, or can be coloured according to whether they match a search word (or

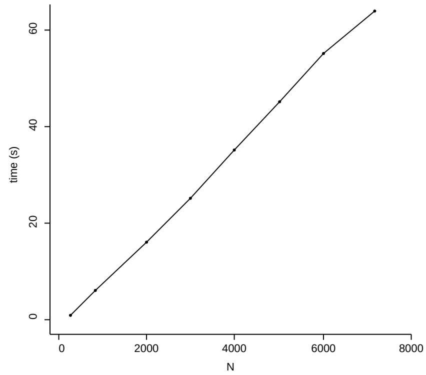

Figure 3. The linear order growth of average iteration time with N is significantly less than the quadratic growth of standard algorithms. The data sets, from the HCI Bibliography, extend to an N of 7135 with an iteration time of 63s and a dimensionality of 20488.

boolean expression) entered via an accompanying Java widget.
The stress falls quickly in the initial iterations, but then the rate of decrease slows. The addition of the flattening force further slows this rate, but allows for lower final stress values than layout processes which operate only in 2D. This characteristic is shared with the standard quadratic algorithm. Layouts of smaller sets have lower stress values e.g. CHI91 with 52 articles has a stress of 0.11, which is (to two decimal places) the same as that resulting from the quadratic algorithm. Below an N of approximately one thousand, stress values are essentially the same. Comparisons with higher N values were not done due to the excessive overall times required for the quadratic algorithm. The linear algorithm starts to show problems with data sets of a few thousand objects, with less coherent clustering and stresses above 0.2..

DISCUSSION

The neighbour sets are a lightweight way to concentrate work in the layout process where it is most needed. With the standard quadratic method, a great deal of work is spent on the separation of objects which are relatively dissimilar. As a first step to reduce this load we adapted the standard algorithm to only exert forces if objects had words in common or were within a threshold distance i.e. distant unrelated objects did not have extra work done on them. This helped the layout process significantly, and was a progenitor of the sampling idea.

In the initial sampling experiments, each object i had a third set involved. This contained those objects for which previous sampling had found a high force. As sampling continued, such 'problem' objects would be pushed or pulled as appropriate until other objects were found to have a higher force. The overhead of maintaining this set was found to be excessive. The many dissimilar objects would jockey for positions in the set, continually entering and leaving it, when all we really wanted to do was be sure that they were all far from i. Exactly how far was not so important, as long as it was 'far enough'. We therefore abandoned the use of the high force set.

In early experiments, we also had higher values for both *Vmax* and *Smax*, for example 20 and 30. It was out of curiosity that we cut these values back to the current 5 and 10. Given that layout quality is still good, one must ask: why is it still good? It is suggested that it works because of indirect linkage via neighbours. Although the object k with the $Vmax+1^{th}$ high-dimensional distance to i will not long be in the neighbour set V_i, it is likely that it will be in a neighbour set V_j where j is a member of V_i i.e. k will be a neighbour's neighbour. Therefore the neighbour will indirectly pull k closer to i. If k and i are directly involved in a random sample at some iteration, then they will be given an extra pull closer.

Tracking the rate of neighbour set insertions shows a very fast initial drop. With the pictured example, the first iteration has 6509 insertions. This number falls to 328 after 12 Runge-Kutta steps i.e. after 3 iterations. After 10 iterations it reaches 77. The figure jitters up and down slightly, but falls to negligible levels after 50 iterations.

We might also mention that a few runs were tried with 831 objects, *Vmax*=0 and *Smax*=15 i.e. purely random sampling. Initially, stress fell very quickly — in fact more quickly than with *Vmax*=5 and *Smax*=10. After the first 56 iterations and a minimum stress of 0.15, when the normal process was still at 0.23, the stress started to rise. It climbed up to 0.18 and slowly fell to a stress value near to that of the layout using neighbours. Although very much preliminary, these results may lead to a hybrid method which takes advantage of this initial speed and then brings in neighbour usage for refinement.

Certain data characteristics have been found to be a significant determinant in layout quality. With the HCI Bibliography, certain years and certain journals or proceedings have large abstracts and also have keyword lists. The latter were found to be especially useful in the layout process. Weighting in the distance metric favours keywords and the more general use of language in abstracts (and in, to some extent, titles) meant that data with a good level of keyword use produces lower stress values than sets where few keywords appear.

Keywords tended to offer better discrimination, giving (as for human readers) a concise set of relatively reliable clues for categorisation or comparison. Older articles in the HCI Bibliography tend to have shorter abstracts and to have keywords less often. When running experiments with larger data sets, programs would inevitably read in many older and less easily discriminated articles. These were harder to lay out.

Another significant factor in generating a low stress value was found to be the exponent p of the scalar product used in the high–dimensional distance metric. Initially we used $p=1$, but found that stress would be reduced significantly when a larger value was used. Again, data discriminability is suggested as being the dominant factor. With a low p, we found that the values for d_{ij} were tightly bunched in a small subrange of the feasible range of [0..1]. Higher values of p tended to widen this subrange. This led to subjectively better layouts as shown by features such as the clustering of related articles and clearer separation of clusters.

Along a related line, we offer a few comments and comparisons based on early experimention with a body of time series data. This data consists of a set of 331 stocks, with each stock represented by a vector of 60 monthly stock price values. Initially we constructed a very simple distance metric akin to the scalar product, based on the proportion of months for which the prices of a pair of stocks would either both rise or both fall. Very recently we have begun experimenting with an information theoretic metric, mutual information. Our tentative results suggest that again discrimination is a key factor. At present, the data set seems to be less cohesively clustered than we would wish. Stress levels reach down to around 0.2, and browsing reveals that although small clusters do appear, global patterns are harder to determine.

It must be admitted that it is harder for an eye untutored in the details of finance and economics to appreciate time series layout subtleties, especially when compared to more familiar bibliographic data. Nevertheless, it is suggested that at present we are not employing a highly discriminating distance metric, and consequently our time series layouts are less cohesive than our bibliographic layouts. Furthermore, we posit that this dependence on

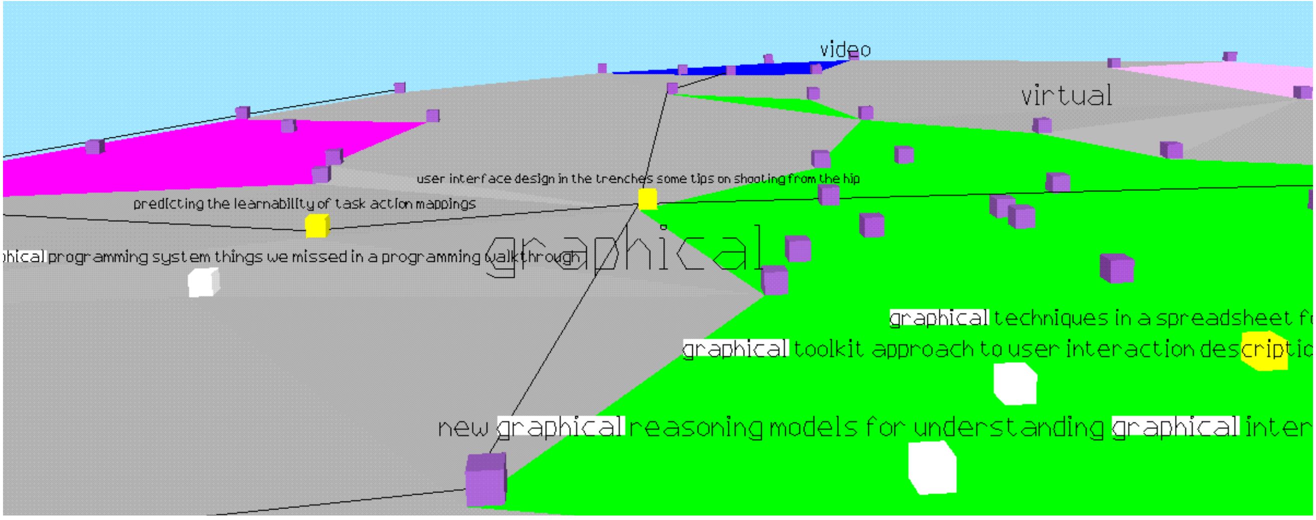

Figure 4. Imageability features build upon basic layout positions. Static features such as coloured districts and paths aid orientation and navigation, as well as describing global features of the data set. In this example, documents are dynamically but randomly chosen to be highlighted with a yellow colour, with a bias based on nearness to the eye. Topic words are also dynamically placed in the scene, based on locality, frequency of occurrence in the field of view, and also word usage history. Topic and title words (e.g. 'graphical') can be clicked on to start searches, which colour hits white.

discriminability is not particular to the presented algorithm but is a performance factor for most layout algorithms that practitioners might bear in mind.

Stress is the metric we use most when looking at layout quality. Others are the maximum force between objects and a histogram of speed distribution. Its correspondence with our notion of a 'good' layout means it is a major focus in assessments. We note, however, two aspects of it.

Firstly, relatively minor changes of stress can lead to significant changes in subjective assessments in layout quality. We notice a strong improvement between 0.16 and 0.14, for example. Another aspect is that dissimilar layouts can have identical (to two decimal places) stress values. This was especially obvious when comparing layouts from an annealing program and a force–based program using the standard $O(N^2)$ technique. Annealing was slightly slower, and tended to lack the consistency and separation of clusters that the force–based program had, even though the two programs led to similar stress values.

These points remind us that subjective assessments of a layout — its 'sense' and 'feel' and ease of use — are the ultimate measure of layout quality. As mentioned in the following section, exploring such human assessments is a topic for future work.

ONGOING AND FUTURE WORK

In the preceding sections, a few comments as to ongoing and future extensions of the presented work were made. We now describe several other directions being followed.

We consider that we should be able to decrease further our minimum stress levels and increase our maximum tractable data set sizes. Occasionally, objects stuck in bad positions are still visible during browsing. One possibility to assist such isolated objects is to selectively add more jitter, as in [11], but we may also try to use the neighbour set to suggest a good place to jump to. If an object's neighbours seem to be staying far away and the object is under stress, then it may be fruitful to jump past intervening objects to where the neighbours are. Also, as mentioned earlier with regard to hybrid algorithms, dynamic variation of *Vmax* and *Smax* values may be useful.

We are experimenting with use of the ongoing structuring process to make the overall layout more efficient. We start with just a handful of objects randomly positioned within a large 3D cube, and alternate between performing one or more steps of the above force/move process and adding in new objects (either randomly or using samples to find 'good' initial locations). We have not yet implemented a scheme for deletions, but refer the reader to [5] for an outline of some approaches.

As mentioned earlier, the stress metric is a significantly complex value to calculate. We have just begun to use sampling methods to approximate the exact stress, and hope to produce reasonably reliable estimates of layout quality at every iteration. Although not directly related to improving layout quality, this would assist experimentation with layout algorithms, most obviously when working on larger data sets. As we start to experiment with time series and other financial data, we hope to learn more of the interrelationship between distance metric, stress and algorithmic effiency.

Another goal of applying this work to financial data is to apply it within UBS, the corporation within which Ubilab operates. We are currently obtaining a body of data with a view to applying our techniques to a 'front desk' situation. If early experiments are successful we will begin a pilot implementation scheme inside the bank this autumn.

Lastly, we briefly mention the information design aspects of our layout work, examples of which are given in Figure 4. Details are the subject of a forthcoming paper, but we are currently experimenting with the addition of static and dynamic features to bibliographic layouts in order to aid navigability and imageability. Static imageability features include the shoreline, local areas of surface roughness and also the colouring of clusters of objects based on geometric proximity (sometimes combined with higher–dimensional proximity). Clusters, paths and landmarks are found using the techniques of [12].

Dynamic features are further subdivided into two types: view–specific and view–independent. Specific to the current field of view are successively sampled words and documents, which are highlighted and have their titles shown. The orientation of text is kept orthogonal to the view of the user as he or she moves. This aids legibility for that user and also helps support awareness of the activities of others inside the shared virtual environment. Independent of the viewpoint are indicators of usage frequency which are shown only on demand. This usage information reflects past activity by all users as measured by mouse selection, keyword search hits, &c., and is updated within the virtual environment as such activities progress. One topic of current interest is the feedback of usage information into the layout process, for example by adjusting term vectors according to the history of use.

CONCLUSION

A layout algorithm has been presented which is suitable for high–dimensional data and which offers significant performance improvements over standard methods for force–based placement. Performance results show a linear order growth in iteration time, without a sacrifice in layout quality. Issues such as data discriminability and the influence of algorithmic parameters were discussed.

The algorithmic work presented here is intertwined with our complementary work on information design and interaction. We have found that apparently high–level design issues such as ease of use and perception are inseparable from low–level issues such as scalar product exponents. Furthermore, we see the need to consider social issues as we start to take advantage of shared environments and usage histories. By treating holistically this range of design factors, it is suggested that we will better reflect the data we visualise in our system and better support the people who use it. As we push on in this direction, we hope that along with our users we will gain further insight and not just numbers.

ACKNOWLEDGMENTS

This work has advanced with the help of Rob Ingram, whose post-doctoral visit led to the clustering features as well as first steps on visualising usage data. Other imageability and Java work was done by Roberto Brega, Christoph Pfranger and Raimond Reichert. My thanks to all of them.

REFERENCES

[1] Card, S.K., G.G. Robertson & J.D. MacKinlay, "The Information Visualizer, an Information Workspace", *Proc. ACM CHI'91* (New Orleans, April 1991), pp. 181–188.

[2] Chalmers, M. & P. Chitson, "Bead: Explorations in Information Visualisation", in *Proc. ACM SIGIR'92 (*Copenhagen, June 1992), published as a special issue of *SIGIR Forum*, pp. 330–337.

[3] Chalmers, M., "Using a Landscape Metaphor to Represent a Corpus of Documents", *Proc. European Conf. on Spatial Information Theory*, (Elba, September 1993). Published as *Spatial Information Theory*, Springer Verlag LNCS 716, A. Frank & I. Campari (eds.), pp. 377–390.

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

[5] Demers, A., et al., "Epidemic Algorithms for Replicated Database Maintenance", *Operating Systems Review 22*, 1 (January 1988), pp. 8-32.

[6] Drew, N., & B. Hendley, "Visualising complex interacting systems", *CHI'95 Conference Companion, Proc. ACM CHI'95,* (Denver, May 1995), pp. 204-205.

[7] Fahlén, L., et al., "A Space Based Model for User Interaction in Shared Synthetic Environments", *Proc. ACM InterCHI'93*, pp. 43–48.

[8] Fairchild, K. S. Poltrock & G. Furnas, "SemNet: Three–dimensional Graphic Representation of Large Knowledge Bases". In R. Guindon (Ed.), *Cognitive Science and its Applications for Human–Computer Interaction*. Erlbaum, 1988.

[9] Feiner, S. & C. Beshers, "Worlds within Worlds: Metaphors for Exploring n-Dimensional Virtual Worlds", *Proc. ACM UIST 90*, 76-83.

[10] Greengard, J., *The Rapid Evaluation of Potential Fields,* ACM Distinguished Dissertation Series, ACM Press, 1988.

[11] Hanson, S.J. , "A Stochastic Version of the Delta Rule", *Physica D 42 (1990), (Special Issue on Emergent Computation)*, pp. 265-272.

[12] Ingram, R. & S. Benford, "Legibility Enhancement for Information Visualisation", *Proc. IEEE Visualization 95*, (Atlanta, Oct. 1995), pp. 209-216.

[13] Lin, X., D. Soergel & G. Marchionini, "A Self–organizing Semantic Map for Information Retrieval", in *Proc. SIGIR'91*, published as a special issue of *SIGIR Forum,* October 1991, ACM Press, pp. 262–269.

[14] MacKinlay, J.D, G.G. Robertson & S.K. Card, "The Perspective Wall: Detail and Context Smoothly Integrated", *Proc. ACM CHI'91* (New Orleans, April 1991), pp. 173–180.

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

[16] Press, W.H., B.P. Flannery, S.A. Teukolsky & W.T. Vetterling, *Numerical Recipes in C (2nd Edition),* Cambridge University Press, 1993.

[17] Robertson, J.D. MacKinlay & S.K. Card, "Cone Trees: Animated 3D Visualizations of Hierarchical Information", *Proc. CHI'91* (New Orleans, April 1991), pp. 189–194.

[18] Rose, K., E. Gurewitz & G. Fox, "Statistical Mechanics and Phase Transitions in Clustering", *Physical Review Letters, 65*, 8, 20 August 1990, pp. 945–948.

[19] Salton, G., *Automatic Text Processing*, Addison–Wesley, 1989.

[20] Wise, J et al., "Visualizing the Non-Visual: Spatial Analysis and Interaction with Information from Text Documents", Proc. IEEE Information Visualization, October 1991, pp. 51–58.