Density Control Through Random Sampling : an Architectural Perspective

Geoffrey Ellis
School of Computing and Mathematics
University of Huddersfield
Queensgate, Huddersfield, HD1 3DH, UK
+44 1484 472912
g.p.ellis@hud.ac.uk

Alan Dix
Computing Department
Lancaster University
Lancaster, LA1 4YR, UK
+44 7887 743 446
alan@hcibook.com

http://www.hcibook.com/alan/topics/random/

Abstract

Information visualisation systems often have to cope with presenting large amounts of data. In this paper we look at the problem of reducing the display density through the use of random sampling. We review some of the uses of sampling and examine density reduction techniques currently used in visualisation. We then look at a proposed novel visualisation that utilises sampling and discuss many issues that arise. Furthermore, we propose the ζDB sampling database architecture and demonstrate how this can be used to enhance current visualisation applications. Finally, we describe the architecture of a sampling aware application and illustrate its advantages.

1. Introduction

In 1546, John Heywood [14] recorded the proverb '*Plentie is no deinte, ye see not your owne ease. I see, ye can not see the wood for trees*', which can be interpreted as, if there are too many trees in the way, one may miss the important information. Imagine that you are walking amongst the trees or just on the edge of the forest and unknown to you there are lots of soldier ants, with potentially dangerous bites, in the vicinity. You can be in danger if you cannot see the routes of the ants because of the number of trees!

This paper looks at the problem of reducing the density of information visualisations, through the use of random sampling. Unlike other techniques that utilise filtering (which may remove the ants altogether from the view), random sampling allows us to see the overall trends in the visualisation but at a reduced density. In the forest scenario, selectively felling 90% of the trees would in fact allow us to see the routes of the ants, although this measure may be unpopular with the forest ranger! Compared with techniques such as filtering or clustering, random sampling requires no prior knowledge of the underlying data and is typically less computationally intensive. As a result, random sampling is ideal for exploratory interactive visualisation.

As well as the obvious use in simulation, randomness has been increasingly used in 'mainstream' applications and algorithms ranging from optimisation, signal-processing, telecommunications and cryptography. However, we shall see that there is surprisingly little use of sampling within existing visualisation algorithms, despite very large data sets being regarded as a 'problem'. This at first seems surprising. The data being visualised is often itself sampled from the real world and so, re-sampling to reduce the data-set size to manageable proportions appears to be a natural extension of this external sampling. However, it seems that the 'throwing away of information' that is inherent in internal sampling from stored databases just seems unnatural to those trained to get the most out of limited data. Indeed, despite the success and importance of stochastic algorithms in many areas, most computing curricula are predominantly focused on deterministic algorithms.

In this paper, we will show that the simplicity and elegance of randomness in computational solutions can equally be applied to information visualisation. In our recent paper [7], we demonstrated how sampling could be used to enhance various standard visualisation and information retrieval techniques including applications which we have previously developed, HiBrowse [9] and Query-by-Browsing [6]. In addition we introduced a novel visualisation, the Astral Visualiser.

Although the advantages of sampling in appropriate circumstances are clear, it also raises some new usability issues, for example, users interpreting sampling artefacts as real trends or patterns. In order to compare user

responses to different solutions we need a flexible sampling-base visualisation architecture to allow different options to be examined. We also want to make sampling part of the standard battery of visualisation techniques. Again this requires a flexible architecture to allow sampling to be used with existing and novel applications.

This paper focusses on the use of sampling to reduce data density in visualisation and the database and visualisation architecture necessary for achieving this. In the next section, we will review some background material. the use of randomness in existing computing applications, ongoing research in sampling from databases and alternative techniques for density reduction in visualisation. Section 3 will then look at random sampling in visualisation including a brief description of the Astral Visualiser and a summary of some of the usability issues it raises. Finally, section 4 considers the requirements for database support and introduce a simple sampling database API, ζDB, to show how it can be used in a sampling-based visualisation architecture and also how it can be constructed as a thin layer using a standard SQL database.

2. Background

2.1. Use of randomness in computer science

Computation may be used to remove or ameliorate unwanted randomness, for example noise reduction in Digital Signal Processing and statistical processing of real world data. However, there are yet more examples where randomness is deliberately introduced into algorithms:

- optimisation and search: genetic algorithms, simulated annealing
- machine learning: neural networks
- setting initial states: hill-climbing, relaxation techniques
- simulation: Monte Carlo methods
- digital signal processing: adding resolution to A-to-D, spreadspectrum & CDM (wireless networks)
- user interfaces: random walks through interfaces
- testing: generating test cases
- telecommunications routing: avoiding overloading 'best' routes
- network protocols: random standoff after contention
- parallel computing: breaking Byzantine cases
- cryptography: generating keys, primality testing, digital signatures
- databases: query optimisation, approximate aggregate results

The use of randomness in solving computational or mathematical problems is often to do with tradeoffs – when an exact solution can never be found (e.g. NP-hard ones) or when it is much quicker to find an approximate solution (e.g. primarility tests, neural networks) or even in symmetry breaking situations (e.g. dining philosophers and narrow-door problems). A comprehensive survey of the use of randomisation in algorithms is presented by Gupta [13] and he asserts that "In an age of rising software complexity and cost, the simplicity of randomized algorithms will be a key determining factor in the acceptance by the software community".

Other reasons for using randomness are based on its very unpredictability. This can help avoid Byzantine situations, such as the systematic failures that brought down the US east coast telephone backbone some years ago, can allow information to be spread over the frequency-range in spread-spectrum techniques, or can simply be hard to guess in the case of cryptographic keys!

For more details on these issues see Gupta's review [13] or our own web page.

2.2. Sampling from databases

If we are going to use randomness to help solve the visualisation problem of too much data then we need to sample our data set, which is typically held in a database. Our recent paper [7], examined some issues regarding sampling from databases and suggested techniques for sampling standard SQL databases. It is important that the statistical properties of the sample are recognised, especially when multiple samples are required. This will be discussed further in section 4.

There has been considerable research in the database literature, since the advent of data warehousing and related data mining application, on the use of sampling in connection with query optimisation. The cost of executing ad-hoc queries on very large databases is considerable, hence the ability to calculate an approximate answer based on a random sample from the database is desirable. Another area of interest in data mining is clustering, where the user wishes to discover distributions and patterns in the underlying data. A technique that uses random sampling and partitioning has been demonstrated to be efficient and accurate even for large databases [12].

Although it is possible to sample from standard SQL databases, this is typically of the order of o(N) where N is the size of the entire database rather than the sample size. It is potentially far more efficient to include sampling primitives into the database manager and thus take advantage of the implementation structure of the database. For example, to obtain a very sparse sample one could randomly choose a small number of blocks and then choose records from these blocks. Efficient strategies have been developed for single joins [4] and also for some aggregate queries [5]. Despite this fruitful research, it

may still be some time before random sampling is a primitive operation in relational databases.

2.3. Density reduction in visualisation

In recent times, more and more information is being gathered, whether in companies information systems or in scientific experiments and consequently the size of databases have been increasing. Information visualisation systems often have to cope with presenting large amounts of data. Various ways of avoiding cluttered visual displays through data density reduction have been devised and some of these can be categorised as follows:

Filtering – In systems using starfield displays [1], slider controls are provided which filter the data on one or more attribute values, thus reducing the amount on the screen. In addition to reducing the density, this may unwittingly remove trend or distribution information depending on the filter values.

Distortion – Fisheye views [11] distort the display so that more screen space is given to the area of interest. The density of this area is reduced at the expense of crowding the peripheral display area. Hyperbolic browser [19] uses a similar technique to deal with hierarchical data. However some users may find this type of distortion disorienting.

Filtering and distortion – A combination of filtering and distortion has been employed in the form of dynamic queries via moveable filters [10] giving the user manual control of the data density on specific areas of the display.

Clustering – Scatter/Gather [22] is one of a number of applications that utilises clustering to reduce the dataset to a manageable density. Clustering reduces the overall density as clusters, rather than individual data items, become the focus of visualisation. Each cluster is typically visualised as a single object, either by showing a representative document (as in scatter/gather) or by some description of the cluster (e.g. feature vector). Some 3D systems such as VR-VIBE [3], also employ clustering to give an overall view of the document space at the expense of individual detail.

Clustering + fisheye – In their Focus+Context system, Kreuseler and Schumann [16] combine a smoothed fisheye view, the Magic Eye View, with dynamic hierarchical clustering of information units to cope with large data sets.

Aggregation – Aggregation is similar to clustering in that it groups data into manageable sets for visualisation. However, the grouping is implicit in clustering, whereas aggregation groups data items based on chosen attributes. Influence Explorer [26] does this by producing histograms so users can see the distribution of data, use brushing to find relationships and utilise the groups (histogram bars) to filter the data. Query previews [8] also makes use of aggregation of attribute values, but the principle objective in this case is to reduce the data to a manageable amount prior to transmission over a network. The user formulates an approximate query with the help of the aggregate data, details are then downloaded and the query can be refined further. The PDQ-Tree Browser [17] provides both an aggregated, hierarchical view of the data set and a tightly coupled detailed view, that allows users to 'prune' the trees using various controls. Irrelevant data is therefore removed thus allowing the users to concentrate on the details of a smaller, hopefully relevant, data set.

One of the problems with visualising information that has not yet been mentioned, is what to display when the user zooms in on an area of the information space. If all the data points are being displayed then the display will become sparse. The two techniques that have been adopted to deal with this problem are: semantic zoom and constant density display.

Semantic zoom – An example is Pad++ [2] which automatically controls the amount of detail present in the display depending on the elevation, z. A similar technique is used in online road map displays [e.g. Multimap] which allows us to zoom in from a map of the UK showing the motorways, to see the major A roads, followed by minor roads and eventually footpaths and perhaps telephone boxes.

Constant information density – A related system employing a form of semantic zoom is VIDA [28] which helps users manually construct applications that maintain a constant overall display density, irrespective of the information space being displayed. These systems preserve the data distribution in x and y dimensions, however this may generate sparse areas in the display. In order to utilise the maximum display area possible, Woodruff et al [29] demonstrate a system of constant information density, where more information is given in areas where there are less data points, thus maximising the display space. Obviously, the actual data density is distorted so care must be taken on the choice of data to display.

Pixelation – An extreme form of semantic zoom is to reduce the data items to a single pixel. Keim [15] has made very effective use of this in high density displays of millions data items. Furthermore, TableLens [23] converts numbers to mini-histogram bars when rows are reduced to single pixel height.

3. Using sampling in visualisation

As far as we are aware, there is little research on the use of sampling as a technique for density reduction in visualisations. References to the use of random sampling occurs in several commercial statistical and data mining applications. For example, Statistica [25] mentions using sampling with very large sample size and Enterprise Miner [24] notes the use of sampling for predictive modelling.

An interesting application of randomisation techniques has been applied to the analysis of spatial data from archaeological studies [18]. The problem there was to overcome the often non-random distribution of data to see statistically valid patterns. Inferential solutions using standard methods were intractable, yet random sampling gave useful approximations.

Olken [21] points out that sampling is used in certain GIS to reduce dataset size prior to graphical display of the data, both with an aim to reduce computational effort and to match the resolution limits of the display.

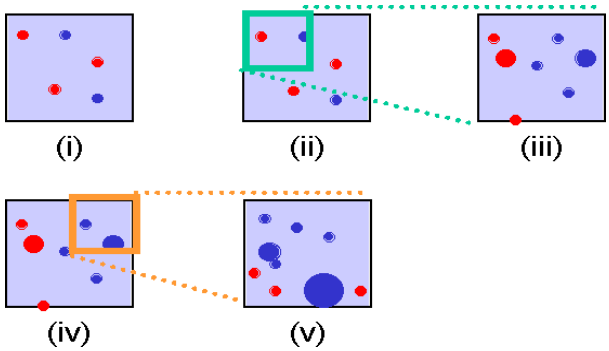
The Influence Explorer [26] also uses sampling as it is based on data from several hundred simulations with random parameters. This is an extreme form of density reduction as their total design space is infinite!

3.1. An example – the Astral Visualiser

In a recent paper [7], we proposed a new sampling-based visualisation technique that is derived from the metaphor of a star-gazers telescope, called the Astral Visualiser which is illustrated in figure 1.

Two attributes or derived values are chosen for the x,y coordinates and a small set of a few hundred sampled records are originally chosen and plotted. The user can select an area of interest and as the Visualiser zooms in, it samples more records in that area (that is a sample constrained by the x and y coordinates). The sample, is chosen so that the density of sampling increases with the square of the zoom value, this means that the actual visible density remains constant (see figure 1). One way of thinking about this, is that we determine the x and y coordinates from the attributes, but randomly allocate a z coordinate and then decide how far away we see by the current zoom factor.

In addition to enabling automatic resampling as the user zooms, a separate 'aperture' control can be applied to alter the sampling density. This is used when the user has panned to an area with much lower or higher density of points. The metaphor is like the aperture in a camera which alters the amount of light reaching the film or CCD – opening the aperture makes the dimmer, more distant 'stars' visible.



(i) initial user view, (ii) user selects top left and zooms, (iii) more points become visible, (iv) selects top right and zooms, (v) yet more points appear

Figure 1. User views of Astral telescope (sizes exaggerated)

3.2. Issues arise from sampling

The concept of using random sampling to control the display density is fairly straightforward but gives rise to a number of issues. We will discuss some here and see [7] for additional interface issues related to sampling.

Zoom in. The metaphor of the Astral Visualiser implies that in order to zoom in by a factor of 2, we need to increase the sampling density by a factor of 4. Generating a given number of new points is not a problem and neither is the distribution, as they are randomly chosen. However we must decide between producing a totally new sample of data points and adding points to the existing sample. Clearly, we need an appropriate sampling method but we must also consider the users perception of the changes in the display. Research has shown [27] that users require landmarks when navigating to develop their spatial knowledge as quickly as possible. So, this suggests that adding points would be the better solution as the existing sample could act as a landmark.

Appearance of new points. Another issue is whether the new points should be displayed differently. In figure 1, the 'old' points are drawn bigger and although this has been exaggerated in the diagram, it is intended to add to the sense of zooming into a starfield. Instead of using size, other visual attributes can be altered such as brightness, colour, or even something more exciting like sparkle or shimmer. It is known [20] that the latter are good candidates for preattentive visual processing, so would be easily 'seen' by the user. Although there are some hints from the literature this is a new technique and so we cannot be sure at this point which are the best visual cues to use.

Zoom out. If a new sample of data points is generated every time the user zooms in, the same process can be applied when zooming out. However, if the policy of adding new points has been adopted, then we need to decide which points should be removed. When zooming back to a previous magnification, the obvious candidates to remove are the ones that were recently added, which implies that some form of history needs to be maintained. But when zooming out to an 'unvisited' magnification, it is more problematic to decide on which points to remove.

Density control. Similar problems as zooming need to be considered when the user changes the global density control. Opening the 'aperture' requires more points, closing the aperture requires less. A new dataset would however be more confusing when changing density as the user does not expect a major change.

Artefacts and coincidence. It is easy to look at random data and see patterns in it that are simply 'artefacts' of the sampling process or coincidental alignments such as lines or clusters. But how can the user know whether these patterns are in fact features or trends. One option is to add a 'reality check' button that resamples the data; so when the users see a potential pattern in the data, they can press the button and if the pattern persists, they can be reasonably sure that it is real. This is the visualisation equivalent of jack-knife techniques used in the statistical modelling and machine learning literature. Note that this will also change the users' visual landmarks so they may become disoriented.

Z-index. We noted that the Astral Visualiser could be regarded as giving data values a randomly chosen z value. In fact, we do actually construct such a random z-index as it simplifies the implementation of many of the issues discussed above. The use of a randomly allocated z-index ensures display inertia when zooming in and also provides the history required whilst zooming out 'for free'. The z-index also makes it easy to ensure continuity if zooming out to an 'unvisited' magnification and when panning. Density control is again simply a matter of filtering on the z-index. Finally, the 'reality check' button corresponds to selecting a fresh z-index. This should be set against the initial cost of preprocessing the z-index information, but our experience with sampling from standard SQL databases shows that this is often necessary (see web page). The notion of a z dimension is used in visualisation tools such as DataSplash [29], which employ semantic zoom techniques, but in these instances, z refers to a layer with different visual attributes (e.g. a point becomes a picture of a tree when viewed at a ‘lower’ level) rather than a sampling attribute, as we propose.

3.3. Non-uniform sampling

Although the Astral Visualiser changes the level of sampling when zooming, this does not depend on the density of the data points but is more a matter of the overall 'view settings' of the user interface. However, in some applications, sampling needs to be more tuned to the location and kind of data.

As an example, consider the data warehouse of a large UK store chain. Sales data is visualised on a map of the United Kingdom. This is achieved by sampling checkout data and then plotting the location of each sale as a dot with colour to indicate product and brand.

Constant density. Constant density systems [28,29] are principally concerned with changing the size of a data point to give more information should there be room on the display. Although this is still possible with our sampling-based visualisation, we have an additional option of increasing the level of sampling in sparse areas in order to maximise the use of the display space. A schematic algorithm would be to repeatedly choose a new data point at random, excluding all points that are within radius r of existing plotted points until there are no suitable points remaining.

This would be the case if we wanted to visualise the sales of particular brands of washing powder across the United Kingdom. There are of course few supermarkets and few sales in less populated parts of the country, so a 'fair' sample would mean it is impossible to see what was happening in the Highlands and Islands of Scotland (too few points) and London would just be an untidy pile. By altering the sampling density based on data density, we can get an even spread of datapoints over the whole country and thus easily spot trends of preferences .

This technique distorts the actual density and so cannot be used if correlations are being sought in the x and y plane. Hence we cannot look for trends in the overall sales of washing powder, we can only look at the relative sales between brands.

Attribute-dependent density. Assume we now want to see if there are patterns relating sales of washing machines and washing powder. There are of course many hundreds more sales of the latter and if we sampled both uniformly we would only see washing powder plotted on our map. Instead, we need to sample the two types of product separately to ensure approximately similar overall numbers of each so that relative differences in different areas become apparent.

Again this technique distorts density and cannot be used to compare actual sales of the two product types, but

it can be used to look for correlations and trends between them.

4. Building sampling-based visualisation

4.1. ζDB database interface

The literature on random sampling from databases emphasises the importance of linking sampling and query operators. Olken [21] argues that sampling operators ought to be embedded within the query processing to reduce the amount of data retrieved and effectively exploit indices created by DBMS. However, there are as yet no standard SQL extensions for sampling.

The simplest sampling interface would be an extension of SQL such as:

SELECT … WHERE … + sample clause

where the sample clause may be: "sample size is N" or "sample size approximately N", or "sample probability p".

Olken [21] also identified several issues for database sampling:

1. the manner in which the sample size is determined
2. whether the sample is drawn with or without replacement
3. whether access pattern is random or sequential
4. whether or not the size of the population from which the sample is drawn is known
5. whether or not each record has a uniform inclusion probability

Some of these have clear answers in visualisation. The sample size (or sampling probability) will be defined as part of the interaction (zooming or density control). The data points are to be plotted or otherwise visualised and hence should normally be drawn without replacement. The entire sample is likely to be used at once, but it may be refined later by the user. The population size is the size of the data set to be visualised, although filtering may mean that the size of the filtered dataset may not be known without querying the database. Finally, we may want to have non-uniform inclusion probability as discussed in the previous section when we considered constant density or attribute-based density.

In addition, visualisation raises some further requirements:

6. the ability to have 'repeatable' samples
7. the ability to chose to resample and have labelled samples
8. the ability to have a randomised output that can be used as a 'z' index.

Note that the 'repeatable' samples refers to the requirement (when we considered zoom in and zoom out) that increasing the sample size includes the original sample. In terms of database operations, this means that the sampling operator has nice algebraic properties. So if S_p is sampling at probability p and F_c is filtering on some condition c, we have properties such as:

if $p \le q$ then $S_p(db) \subseteq S_q(db)$

$S_p(F_c(db) \cup F_{c'}(db)) = S_p(F_c(db)) \cup S_p(F_{c'}(db))$

…

We have found that the inclusion of one or more generated random z-index fields is not only useful for visualisation itself, but also makes it easy to satisfy these quite stringent conditions. So, if the z-index is chosen in the range [0, Zmax-1], we can translate "with probability p" into "z less than p*Zmax" etc. Repeated sampling based on the same z-index will be identical (requirement 6) Different z-indices correspond to different labelled samplings (req. 7). Also, since the named z-index fields are regarded like other fields for selection purposes they can be included in the query result (req. 8). Furthermore, the selection condition on the z-index can be varied depending on other field values, thus allowing non-uniform sampling.

The resulting database interface therefore has two main functions:

sample(table, field_list, condition, prob, z-index-name)

generate_z(table, z-index-name)

Note that field_list may include named z-indices and prob may depend on field values.

We refer to the resulting sampling database as ζDB (zetaDB). We have worked with this as a thin implementation layer over a standard SQL database, although it would be better to use a database with sampling as primitive when these become available.

4.2. Non-sampling visualisation architecture

Before describing the architecture of our sampling aware applications, we will first look at a typical visualisation architecture as shown in figure 2.

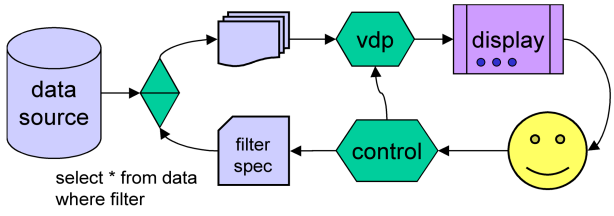


Figure 2. Typical visualisation architecture

In order to gain an insight into the data being presented, users alter some controls (e.g. slider, checkbox, option list). These actions send commands to the visualisation/graphics display processor (vdp) for

operations such as pan, local zoom or distortion, and can also make changes to the database filter (e.g. update an SQL command) which will result in the local data set being updated. The user receives visual feedback.

4.3. ζDB architecture

Figure 3 shows a generalised view of the ζDB sampling database management system. Its function is to receive sample requests and return this data together with a z-index which can be used by the visualisation controller for local processing.

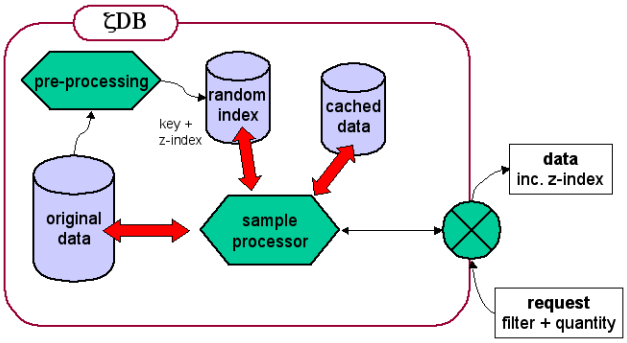


Figure 3. Sampling database architecture

The 'sample' request generates a named random index (z-index). In our SQL wrapper, this is simply implemented by either adding a random column to the original table or by creating a special index table of key–z-index pairs.

The 'sample' request includes the items of data required and filter conditions (if any) together with a data quantifier (typically a proportion or number of samples) and specified z-index. This is converted into an SQL request using the z-index for sampling.

Our SQL wrapper does not include any caching of results over and above those of the underlying database, but this could be added to reduce database load, especially for small samples from very large or networked datasets.

4.4. Sampling-based visualisation architecture

Sampling-based visualisation using ζDB can be achieved in two different ways, firstly, by adding sampling to an existing visualisation application and secondly by developing a sampling-aware application. We will now consider each of these in turn.

Loosely coupled visualisation. In this option, we take an existing visualisation and simply substitute the full data set in the data source (in figure 2) with a sample from the ζDB (see figure 4).

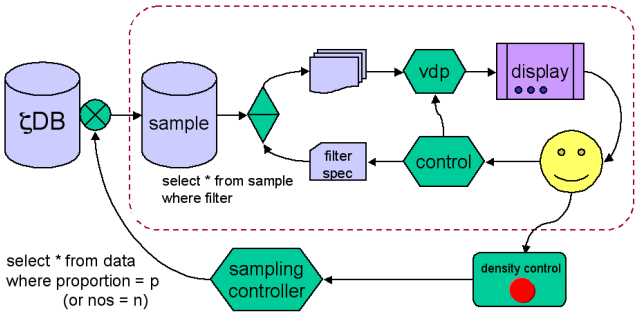


Figure 4. Sampling architecture, with separate density control

A separate density control can be added and manipulated by the user to increase the sampling rate. The sampling controller takes this and instructs ζDB to generate more or less data points. This function is useful in applications where there are just too much data to display. The original application needs to know nothing about the sampling except that it must know when its data has changed in order to refresh the visualisation.

Note how filtering and sampling are separated in the diagram. The original application controls the filtering whilst the external density control changes the sampling.

We could nearly implement Astral Visualiser in this way by taking a simple 2D scatter plot with zoom and pan as the base system. But such a loosely coupled architecture would not allow the density to be automatically increased as the user zooms into a region.

Tightly coupled visualisation. When the application needs to directly control the sampling in tandem with the filtering (as with the non-uniform density examples discussed in section 3.4), a more tightly coupled architecture is required, as shown in figure 5.

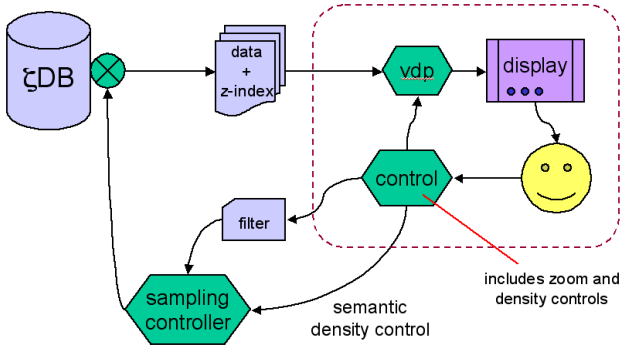


Figure 5. Sampling aware application architecture

Using the Astral Visualiser as an example, the user has a zoom control, as found in many visualisation applications, and also a data density control. As the user zooms in, the application increases the sample probability and changes the filter. The sampling controller therefore

instructs ζDB to generate more data points in the new display area so that the overall density of the display remains constant. Note also how the z-index becomes part of the resulting data set and so it can be used in the visualisation (as described in section 3.3).

The density controls are now integrated into the application interface and all filtering and density functions are handled by the sampling controller. As discussed earlier, filtering and density are related and their functions can therefore be optimised by one controller. Similarly, the ζDB can optimise the sampling, thus making a separate sample database redundant.

The two store chain scenarios from section 3.4 can be implemented over this architecture. In the case of washing machines vs. washing powder, the visualisation application simply sets the sampling probability to be an SQL 'if expression' with value dependent on the product type.

In the case of the different washing powder brands where we wanted constant density, this is a little more complicated as the final level of sampling has to be done in the application. The latter must divide the display area into a reasonably small number of sections over which the raw density does not vary too much. These sections can then be given different probabilities when sampling the database to give a near uniform, but deliberately slightly over-dense distribution. Lastly, the application has to perform the final selection based on the schematic algorithm in section 3.4, choosing at each stage the data with the lowest z-index.

Note how significant performance benefits can also be gained from associating a z-index with each data item. Without a z-index, ζDB would need to determine the new data set whenever the user zooms in or increases the density. Instead, with the z-index, the local visualisation application can change the upper bound of z until the required density is achieved. The application can also sub-sample based on the z-index knowing that this will yield the same answers as the ζDB request. In the case of constant density, this is particularly important as the conditions required 'not within radius r of previous items' would be extremely inefficient or impossible to implement through SQL queries.

5. Summary and future work

In previous work we have argued that random sampling is a potentially powerful method to use in visualisation. In this paper we have built upon this, looking particularly at the issue of density reduction for large data sets.

We have described the Astral Visualiser, a novel visualisation technique that uses sampling as a fundamental part of its interaction. We have also discussed other scenarios where sampling can be used. As a result of our practical experience and analysis of scenarios, we have proposed a sampling database interface ζDB (currently implemented as a thin layer over an SQL database) and two visualisation architectures, one suited as a loosely coupled 'bolt on' for existing visualisations and the other for situations, such as the Astral Visualiser, where the sampling is more tightly coupled to the visualisation control.

Of particular importance in ζDB is the use of named z-indices that enable us to ensure repeatability in sampling or force different samples where required. As well as being useful for ensuring appropriate properties of samples, the z-index can be made available to the visualisation itself, as is the case in the Astral Visualiser.

There are many interesting user-interface issues raised by sampling and one reason for the construction of a generic sampling-based visualisation architecture is to make it easier to study visualisation options as part of our ongoing work. Furthermore, going beyond the visualisation of flat data structures and looking also at relationships, there are yet more issues, for example, if we have a uniform sample we are likely to underestimate inter-relationship.

Even before the results of this more systematic study we believe that in many situations sampling can already be an effective tool in the visualisation of large datasets.

6. References

[1] C. Ahlberg and B. Shneiderman, "Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays," *Proc. CHI'94*, ACM Press, Boston, April 1994, pp. 313-317.

[2] B.B Bederson and J.D. Hollan, "Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics," *Proc. UIST'94*, ACM Press, Marina del Rey, CA, Nov 1994, pp. 17-26.

[3] S. Benford et al. "VR-VIBE: A Virtual Environment for Co-operative Information Retrieval," *Proc. Eurographics 95*, Blackwell Publishers, 1995.

[4] S. Chaudhuri, R. Motwani and V. Narasayya, "On Random Sampling Over Joins," *Proc. SIGMOD '99*, ACM Press,1999, Philadephia, pp. 263-274.

[5] S. Chaudhuri, G. Das and V. Narasayya, "A Robust, Optimization-Based Approach for Approximate Answering of Aggregate Queries," *Proc. SIGMOD '01*, ACM Press, Santa Barbara, CA, May 2001.

[6] A. Dix and A. Patrick, "Query By Browsing," *Proc. 2nd International Workshop on User Interfaces to Databases (IDS'94)*, Springer Verlag: Workshops in Computer Science, Lancaster, UK, April 1994. pp. 236-248.

[7] A. Dix and G.P. Ellis, "By Chance: Enhancing Interaction with Large Data Sets Through Statistical Sampling," to be published in *Proc. AVI '02*, Italy, May 2002.

[8] K. Doan, C. Plaisant, B. Shneiderman and T. Bruns "Interface and Data Architecture for Query Preview in Networked Information Systems," *HCIL Technical Report No. 97-09*, http://www.cs.umd.edu/hcil, 1997.

[9] G.P. Ellis, J.E. Finlay and A.S. Pollitt, "HIBROWSE for Hotels: Bridging the Gap Between User and System Views of a Database," *Proc. 2nd International Workshop on User Interfaces to Databases (IDS'94)*, Springer Verlag: Workshops in Computer Science, Lancaster, UK, Apr. 1994. pp. 45-58.

[10] K. Fishkin and M.C. Stone, "Enhanced Dynamic Queries via Moveable Filters," *Proc. CHI'95*, ACM Press, Denver, May 1995, pp. 415-420.

[11] G. W. Furnas, "Generalized Fisheye Views," *Proc. CHI'86*, ACM Press, Boston, Apr. 1986, pp. 16-23.

[12] S, Guha, R. Rastogi and K. Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," *Proc. SIGMOD '98*, ACM Press, Seattle, June 1998, pp. 73-84.

[13] R. Gupta, S.A. Smolka and S. Bhaskar "On Randomization in Sequential and Distributed Algorithms," *ACM Computing Surveys*, Vol. 26, No. 1, Mar. 1994.

[14] J. Heywood, *A dialogue Conteynyng the Nomber in Effect of all the Prouerbes in the Englishe Tongue*, 1546

[15] D. A. Keim, "Pixel-oriented Visualization Techniques for Exploring Very Large Databases," *Computational and Graphical Statistics*, vol. 5, no. 1, 1996, pp. 58-77.

[16] M. Kreuseler and H. Schumann, "Information Visualization Using a New Focus+Context Technique in Combination with Dynamic Clustering of Information Space," *Proc. New Paradigms in Information Visualization and Manipulation (NPIV'99)*, Missouri, November 1999, pp. 1-5.

[17] H. Kumar, C. Plaisant and B. Shneiderman, "Browsing Hierarchical Data with Multi-level Dynamic Queries and Pruning," *Human-Computer Studies*, 46, 1997, pp. 103-124.

[18] K.L. Kvamme, "Randomisation Methods for Statistical Inference in Raster GIS Contexts," Dept. of Archaeology & Center for Remote Sensing, Univ. of Boston, USA.

[19] J. Lamping and R. Rao "Visualizing Large Trees Using the Hyperbolic Browser," *Proc. CHI'96*, ACM Press, Vancouver, Apr. 1996, pp. 388-389.

[20] K. Nakayama and G.H. Silverman, "Serial and Parallel Processing of Visual Feature Conjunctions," *Nature*, 320: 1986, pp. 264-265.

[21] F. Olken, *Random Sampling from Databases*, Ph.D. dissertation, UC Berkeley, LBL Technical Report 32883, April 1993.

[22] P. Pirolli et al., "Scatter/Gather Browsing Communicates the Topic Structure of a Very Large Text Collection," *Proc. CHI'96*, ACM Press, Vancouver, May 1996, pp. 213–220.

[23] R. Rao and S. Card, "The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabular Information," *Proc. CHI'94*, Boston, ACM Press, 1994, pp. 111–117.

[24] http://www.sas.com/products/miner/

[25] http://www.statsoftinc.com

[26] L. Tweedie, R. Spence, H. Dawkes and H. Su. "Externalizing abstract mathematical models." *Proc. CHI'96*, ACM Press, 1996, pp. 406–412.

[27] N.G. Vinson, "Design Guidelines for Landmarks to Support Navigation in Virtual Environments," *Proc. CHI'99*, ACM Press, Pittsburgh, May 1999, pp. 278-285.

[28] A. Woodruff, J. Landay and M. Stonebraker, "Constant Information Density in Zoomable Interfaces," *Proc. AVI'98*, ACM Press, L'Aquila, Italy, pp. 57-65.

[29] A. Woodruff, J. Landay and M. Stonebraker, "Constant Density Visualizations of Non-Uniform Distributions of Data," *Proc. UIST'98*, San Francisco, 1998, pp. 19-28.