Incremental Board

A grid-based space for visualizing dynamic data sets

Roberto Pinho
Universidade de São Paulo
Caixa Postal 668 São
Carlos,SP 13560-970 Brazil
robertopinho@acm.org

Maria Cristina F. de Oliveira
Universidade de São Paulo
Caixa Postal 668 São
Carlos,SP 13560-970 Brazil
cristina@icmc.usp.br

Alneu de A. Lopes
Universidade de São Paulo
Caixa Postal 668 São
Carlos,SP 13560-970 Brazil
alneu@icmc.usp.br

ABSTRACT

In Information Visualization, adding and removing data elements can strongly impact the underlying visual space. We introduce a chess board analogy for displaying (projecting) objects from a dynamic set on a 2D space, considering their similarity in a higher dimensional space. Our solution is inherently incremental and maintains a coherent disposition of elements, even for completely renewed sets. The algorithm considers relative positions, rather than raw dissimilarity. It has low computational cost, and its complexity depends only on the size of the currently viewed subset, V. Thus, a set of size N can be sequentially displayed in $O(N)$ time, reaching at most $O(N^2)$ only if viewing the whole set at once. Consistent results were obtained as compared to (non-incremental) multidimensional scaling solutions. Moreover, the corresponding visualization is not susceptible to occlusion. The technique was tested in different domains, being particularly adequate to display dynamic corpora.

Categories and Subject Descriptors

H.5.m [**Information Interfaces and Presentation**]: Misc.

General Terms

Algorithms

Keywords

High-dimensional data visualization, Multidimensional scaling, Projection

1. INTRODUCTION

One option to visualize high-dimensional data sets is to layout their elements on a 2D space according to the similarity among items. This solution has been adopted to explore document collections [11, 27, 31], to display thumbnail collections of images [6, 28, 29] and to visualize general multidimensional data stored in databases [4].

SAC'09 March 8-12, 2009, Honolulu, Hawaii, U.S.A.

However, if one wishes to visualize a dynamic set where objects are constantly being added and removed, she/he can either (i) rely on a layout fixed a priori, which would not capture major changes in data, or (ii) redo the whole map at critical steps, which might be disturbing to users as new map layouts may bear little or no resemblance to the original layout. We propose here a third strategy that incrementally positions and re-positions data items on a chess board-like space as elements are added and removed while preserving the overall layout configuration.

If we were to add chess pieces to a fixed position on a chess board, the arrival of a new piece would cause one or more pieces to be displaced to accommodate the new one. Having a goal of placing similar pieces together, we could steer the way we displace pieces in order to achieve that goal. The problem could also be seen as one of sorting elements by similarity, but having two dimensions to work with, resembling somehow a 2D expanded insertion sort. When a new item arrives in an insertion sort, it may displace neighboring items, whilst the size of the allocated space increases.

A potential application of such a technique is to track changes in scientific literature. It would enable users to add new articles to maps that they are familiar with, maps that could also have been fine tuned by removing undesired papers. When new articles arrive, their location and impact on the map could provide these experienced users direct clues about their content and trends in research. This would not be possible if the choice is to redo the map or follow earlier layouts. In any of those scenarios, the arrival of new elements can have an impact on the underlying visual space. For instance, as a new research topic becomes popular, it could fill the map, squeezing together some loosely related articles, thus the meaning of neither absolute positions nor distances is consistent across different time moments, a fact that can be very misleading to a human analyst. On following these examples and analogies, we can not rely on fixed visual axes, nor in absolute distances among elements. The important factor should rather be where elements are placed in relation to one another.

Based on the above rationale, we consider relative similarity of elements in the original high-dimensional space to place (project) them on a 2D space that follows a chess board analogy. Our incremental strategy allows the effective display of dynamic data sets. The resulting visual space maintains a coherent disposition of elements from constantly updated sets, even though it does not rely on fixed, well defined or explicit visual axes or dimensions (as illustrated in Section 5). Moreover, even if no item from the final set has

co-existed with those from the initial set, their placement still follows the same global relative disposition of classes or categories as in the initial set of a given collection. A specific visualization based on the board analogy was developed as a proof-of-concept and is introduced in this paper.

We compared our approach with other multidimensional scaling strategies and obtained very competitive results even when working with static data sets (see Section 5). We have tailored our approach to provide a specific visualization that handles ever changing corpora, one from which documents may be added, removed or replaced, which is not supported by current placement techniques. We have also provided a image thumbnail collection visualization example that benefits from the fact that occlusion never occurs on our board-like space, where only one element can occupy each cell.

The following section briefly describes related techniques and potential applications. Section 3 (i) details the board analogy space, (ii) presents how distances may be calculated over it, (iii) describes how elements are added, removed and replaced from the board, and (iv) discusses algorithm complexity. The Incremental Board visualization is presented in Section 4. Section 5 brings case studies used to compare the proposed approach using quantitative measures and used to explore possible applications of the technique. Finally, conclusions and further work are discussed in Section 6.

2. RELATED WORK

Visualizing document collections is a typical application of high-dimensional data visualization. Document collection visualizations are implemented in systems such as CiteSpace II [11] and the Projection Explorer – PEx [27]. On those systems and on other related knowledge domain visualizations [8, 10], typically there is no conceptual and/or explicit meaning assigned to each visual dimension. The same can be said about the FastMapDB tool [4], which uses 2D and 3D layouts to display collections of structured data from databases. Notable exceptions are the explicit mapping of height in 2.5D representations (e.g. [21]), scatterplot-based views, and, on some systems, the mapping of time to one of the visual axes. In these latter examples, only a few of the original data dimensions are explicitly mapped to a visual dimension, not contemplating a true solution for n-dimensional data. Some systems, such as PNNL's IN-SPIRE [31], apply techniques that rely on extraction of factors, such as Principal Component Analysis (PCA) and Latent Semantic Analysis (LSA) to layout documents. They might be able to represent more than two or three of the original dimensions on a two-dimensional space, but they map only the first few significant or "latent" dimensions. Hence they are still inadequate for handling high dimensional data [26].

Nonetheless, in any of those solutions, visualizing changes in data requires building new layouts, or, in some cases, the use of visual dimensions established a priori from an initial set, which may be inadequate to layout latter versions of the data set.

One solution to this problem, as employed by Chen in his Citespace II visualization [11], is to build the map beforehand using the whole data set and then manipulate some visual attribute (for example, transparency) to add or remove elements from the representation. Although this solution gradually presents the evolution of a data set while maintaining a consistent layout of elements, it cannot be considered as truly dynamic, since the complete collection must be fully available before the visual representation can be constructed. The map is not incrementally built as new documents are added to the corpus, removed from it, or yet when existing documents are replaced with new ones.

Our grid-based space immediately reminds the use of self organizing maps (SOMs) for visualizing document collections [14]. However, those results bear major distinctions to ours: (i) grid dimensions are fixed beforehand, (ii) a single cell is allowed to hold multiple documents or elements, and (iii) once elements share a cell, similarity relations can only be inferred for the cell, relations among individual elements are not represented. Similarly to our approach, SOM's algorithm complexity is a function of the number V of map units (grid cells), not of the size N of the data set [13]. Nevertheless, to display a large data set using a small grid size V, a SOM would fit all elements into the available cells, whereas our approach displays them sequentially, having V elements on display at any given time, which is more suitable for dynamic and time stamped data sets.

Dynamic extensions to SOM have been proposed, such as Incremental Grid Growing(IGG) [7], and the growing self-organizing map (GSOM) and subsequent developments [1, 2]. They overcome the requirement for pre-defining grid dimensions, nonetheless many data items may still cluster together in a single cell (neuron), while other cells remain empty. This feature is useful to provide high level visualizations or to build hierarchical solutions, but at the expense of not presenting individual elements. An undesired feature of IGG and GSOM, also pointed out elsewhere [24], is that they grow only from the border, leaving high density areas (many elements in each cell) at the center. Results from IGG seem to degrade when growing branches or arms merge. Latter versions of GSOM try to tackle this problem, but require smoothing phases that increase the complexity of the solution. Description of a removal process for neither GSOM or IGG has been found hitherto.

Chalmers briefly mentions that new elements might be added while his multidimensional scaling (MDS) technique converges to a solution, though this possibility is not further explored [9]. Once the projected data set becomes stable, he expects that reaching a solution will require a few $O(N)$ iterations of his stochastic force directed placement procedure. Latter enhancements to the technique also require a fine-tuning phase to reach the final solution [23]. In either case, complexity would lie well over $O(N^2)$ in an incremental scenario.

A more robust solution is presented by Law & Jain for an incremental ISOMAP algorithm [18]. However, coordinates for all data items must be updated and can change dramatically after each element addition [19]. Besides requiring extra computation to update the whole set, a global rearrangement of coordinates, as named by Law et al., is not desirable for our goals. In contrast, our solution requires few changes on coordinates for each added (or removed) element (see Section 6).

Basalaj's Ph.D. Thesis [5] introduces an incremental multidimensional scaling procedure. However, his approach requires a previously computed minimal spanning tree for the data set, thus undermining its application on dynamic data sets. As we do, he also advocates the use of grid-like visualizations of elements arranged by similarity, the so-called Proximity Grids. He introduces some options on how to

build a grid from a final MDS layout. His *bump* approach is somewhat similar to our element addition procedure, although it does not consider the re-arrangement of the existing elements to reflect the changes introduced by the arriving ones. It is thus not appropriate for incremental building, being dependent on a previously built layout.

Related studies by Rodden et al. [28, 29] show some evidence that arranging images by similarity is useful to designers searching for photographs and also that avoiding overlap is desired by users. These principles have been applied in the Photomesa application [6] to display clusters and hierarchically organized image collections using treemaps, and a simpler layout for image clusters called bubblemaps. Both goals of layout by similarity and occlusion avoidance can be accomplished by our technique, with the additional benefit of being able to gradually update a visualization. A possible application of our solution is thus to maintain a personal image library that gradually receives additional images and yet keeps a layout that the user is familiar with.

3. THE INCREMENTAL BOARD SPACE

The Incremental Board(*incBoard*) space is a grid-based 2D space with cells arranged in rows and columns. It is expected that only a single element should occupy a cell at any given moment, except when adding items. Formally:

An element E_i placed on this board space is represented by a point p_i with coordinates (x_i, y_i), with $x_i, y_i \in \mathbb{Z}$. In other words, $p_i \in \mathbb{Z}^2$.

The incremental board space assumes two possible states: (i) a stable state, where no two elements share the same cell coordinates and (ii) an unstable state, where two or more elements share a single cell. The unstable state requires action in order to bring the board back to a stable condition.

The distance between two elements over the board is calculated using the Chebyshev distance $d_c(E_i, E_j)$:

$$d_c(E_i, E_j) = max\{|x_i - x_j|, |y_i - y_j|\} \quad (1)$$

The Chebyshev or chessboard distance considers the number of steps required to move from one cell to another. On the chess board analogy, it reflects the movement of the king.

The placement of elements on the 2D board could reflect their relative positioning on the original n-dimensional space or their relative ranking derived from some dissimilarity measure evaluated between elements. So, for each element E_i on the board an error can be calculated to reflect the difference in the ranking of other elements when considering their distances on the 2D space $d_c(E_i, E_j)$ as compared to their dissimilarity $\delta(E_i, E_j)$, defined in the original n-dimensional or conceptual space.On the hypothetical example presented in Table 1, the error should be zero, as the sorting of other elements relative to E_1 would result in the same ordered set $\{E_3, E_2, E_4\}$, using any of the distances and ranks shown.

Table 1: Relative position ranking example. $R_{c1}(E_j)$ is the ranking on the 2D space relative to E_1. $R_{n1}(E_j)$ is the ranking on the nD space relative to E_1.

E_j	$d_c(E_1, E_j)$	$R_{c1}(E_j)$	$\delta(E_1, E_j)$	$R_{n1}(E_j)$
E2	5	2	1000	2
E3	4	1	850	1
E4	6	3	5000	3

We weight the error measure both by: (i) the difference from the expected position $(R_n i)$ to the actual position $(R_c i)$ of each of the other elements, and (ii) the expected position $(R_n i)$, assigning more weight to errors originating on the vicinity of the reference element E_i.

For a list L of elements, the weighted error $W_{err}(E_i, L)$ relative to an element E_i is given by:

$$\sum_{E_j \in L} |R_{ci}(E_j) - R_{ni}(E_j)| \times (|L| - R_{ni}(E_j)) \quad (2)$$

Early experiments showed that the above error measure often yields the same value for different placement options (see Section 4). If this is the case, we use a weighted error count $C_{err}(E_i, L)$ as a second criterion:

$$\sum_{E_j \in L} \begin{cases} |L| - R_{ni}(E_j) & \text{if } |R_{ci}(E_j) - R_{ni}(E_j)| \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Having defined these error measures, we can pursue the "goal of obtaining a monotone relationship between the experimental dissimilarities or similarities and the distances in the configuration[projection]", as stated by Kruskal [17] for nonmetric multidimensional scaling (MDS).

3.1 Dynamic Construction

With the exception of the first element, new elements are always added, one at a time, at the cell that currently holds its most similar element. Then, as the chosen cell becomes unstable, a process is started to bring the board back to a stable state. The first element is simply positioned at any location.

The process of finding the most similar element can be costly, so the system operates in two possible modes: (i) full and (ii) stochastic sampling.

In full mode, the arriving element is compared to every element on the board, whilst in the stochastic mode a fixed list of close neighbors is used instead. The stochastic sampling mode is derived from Chalmers et al. [9]. Besides the list of neighboring elements, a list of randomly selected elements is also kept. These latter two lists are used to update the weighted error measure when elements are moved during the addition or removal processes detailed below.

For a cell temporarily holding two elements, 16 solutions are evaluated to bring it back to a stable state. Eight of them keep the new element at the center cell and move the old one to one of its 8 neighboring cells, and 8 alternative solutions keep the old element at the center and move the new one. The one option that introduces the lowest added error for both elements is chosen.

Eventually, the displaced element will fall on another already occupied cell, in which case the process is repeated with the two elements sharing this new cell. In order to avoid cycles, a list with already visited cells is kept and these are not considered again as an option. The process ends when a moved element falls on an empty cell.

If all neighboring cells of an unstable cell have already been visited, the disturbing element is trapped and a special greedy procedure is applied to move it until it finds a non-trapped cell. At each step, the trapped cell's neighboring cells are evaluated and the one associated with the minimum error is chosen. Rows and columns left behind on the previous step are not considered any more. For example,

if the element is moved from row 5 to row 6, only rows $[6, \infty[$ are considered in the future. The procedure stops when a cell with non-visited neighboring cells is found. Then, the regular process can resume. The list of visited cell is always cleared before adding a new element.

When removing elements, the board layout is adjusted pursuing two goals:(i) better reflecting relative positions once an element has been removed, and (ii) keeping the board as dense as possible, avoiding scattered elements. So, we try to move elements closer to the center of the board. If the just emptied cell is above and to right of the center, we fill it with an element chosen from one of the 3 cells above or to the right of it. The option which yields the lowest added weighted error is chosen. The process is repeated with the newly emptied cell until all the 3 options are empty cells.

The most costly operation when adding or removing elements is to update the error measures for all the positioning alternatives considered. To handle this effectively, the process can select between operating in the full mode or in the stochastic sampling mode.

The full mode simply uses the currently list of elements on the board B as the list L to compute weighted errors. This is a suitable option on maps with a low number of elements, or when the data set rarely changes. Using the full mode, we were able to obtain satisfactory performance when handling up to 400 elements simultaneously on the board, with performance degrading rapidly after that mark (see Section 3.2).

In the stochastic sampling mode, weighted error calculations for each element consider a close neighbors list and a random list of elements, both having fixed size, as indicated by Chalmers et al. [9]. Both lists start empty and are filled with randomly selected elements, with the ones most similar to the reference element kept in its close neighbors list. Every time an element is a candidate for movement, its lists are updated, clearing and repopulating the random list. The neighbors list is updated if any of the newly selected random elements is closer to the reference element than those previously on its neighboring list. The list of close neighbors of a newly added element E_i, LN_{E_i}, is also improved using the list of close neighbors of its current closest neighbor E_j, LN_{E_j}. This improvement process is repeated if a different closest neighbor is found.

The mode to adopt may be chosen in real-time. For dynamic data sets, the decision is based on the rate of new elements received, whereas for static data sets the choice is based on the elapsed time required to add new elements. Ideally, one should use the full mode to add the initial hundreds of items, and switch latter on to the stochastic mode. This mode would then work over a more stable layout, where the random choice of a particular item does not significantly impact the layout, while keeping the computational cost low.

3.2 Algorithm complexity

All required operations consider only those elements currently on the board. Therefore, a whole data set with size N can be sequentially displayed with a constant cost for adding and removing elements, as long as the viewing window (board size, or number of occupied cells) has a fixed size V. The overall complexity would be $O(V \times N)$ or simply $O(N)$, as V remains constant.

A particular case, which we further analyze here, is found when the final board size V matches the data set size N, thus producing a map of the whole data set.

Using the stochastic sampling mode, each movement (selecting which of the two elements in a shared cell should move and where to) requires a constant effort k, proportional only to the size of neighbors and random lists. For each added element, a certain number of movements is required until a moving element reaches an empty cell. An alternative view of the process is to imagine that an unstable cell state is moving towards an empty space. If the board is evenly occupied (close to the shape of a square), the maximum distance from the initial unstable cell to the border is $\sqrt{V}$. It means that, the process should stop in $\sqrt{V}/2$ movements. Under this ideal behavior, the overall complexity would be $O(N \times \sqrt{V}/2 \times k)$ or $O(N^{3/2})$, when displaying the whole data set at once ($N = V$).

However, the space is not always evenly used and the movement of the virtual unstable cell is not constrained to go towards the nearest edge, being free to wander on the board as long as it does not move to any previously evaluated space. So theoretically, in the worst-case scenario, all cells could be visited every time, bringing the complexity to $O(N \times V \times k)$. Thus, $O(N^2)$ is the worst possible complexity, considering the particular case where the viewing window size equals the size of the data set.

Henceforth, the algorithm's complexity, when $N = V$, should lie between these two limits: $\{O(N^{3/2}), O(N^2)\}$. An empirical analysis using a corpus of 675 scientific articles showed that the number of required movements stayed well below V for each element addition.

4. THE INCREMENTAL BOARD VISUALIZATION

The incremental board analogy enables a visualization without occlusion and with fixed screen space allocated to each data item. Sophisticated glyph models can be designed to represent individual elements at multiple zoom levels. For fewer elements, the glyphs may carry the element's name, size or any other available information (see Section 5.2 for an example displaying image thumbnails). They could, for example, display a miniature of a document's front page. To display more elements, smaller rectangular glyphs may be chosen.

Delimiting cells with grid lines is optional, as hiding the grid may result in a cleaner presentation and improved visualization.

Displaying newly added elements is an important component of the visualization, as is animation to help users tracking layout changes. Two viewing options are available: (i) a step mode, where the process is paused after each element addition, removal or replacement (a new element replaces an existing one), and (ii) a continuous mode, with no pause between operations.

A time or item count sliding window is available for viewing time sensitive data. So older elements are removed once their age is above a threshold or the item count grows above the limit. Glyph transparency may reflect element age, for instance, documents may be made to fade away prior to being actually removed from the visualization.

5. CASE STUDIES

We present three of the case studies conducted to evaluate the incremental board space and the corresponding visual-

ization. The first two case studies compare incBoard with other MDS techniques using quantitative measures. The last one uses the developed proof-of-concept application to explore an additional potential usage scenario of our technique, displaying image thumbnails.

5.1 Comparison with MDS techniques

We have used stress [15, 16, 17] as quantitative measure to evaluate incBoard as a MDS technique. In the original stress equation (4), $\hat{d}(E_i, E_j)$ is a fitted distance dependent on the dissimilarity $\delta(E_i, E_j)$. A simplified version of stress has been previously used to evaluate projection techniques [9, 26] that assumes $\delta(E_i, E_j) = \hat{d}(E_i, E_j)$. As we favor relative positions, rather than trying to match expected distances (that is, we are dealing with a nonmetric MDS process), such an assumption does not hold and we must stick to the original hypothesis that $\hat{d}(E_i, E_j)$ is an unknown monotone distortion of $\delta(E_i, E_j)$ [17]. Moreover, distances in *incBoard* are in the range of $[1, \infty[$, while dissimilarities often fall in the range of $[0, 1]$, henceforth an exact match between $\delta(E_i, E_j)$ and $d_c(E_i, E_j)$ is not expected.

$$stress = \sqrt{\frac{\sum_{i<j} \left(d_c(E_i, E_j) - \hat{d}(E_i, E_j) \right)^2}{\sum_{i<j} d_c(E_i, E_j)^2}} \qquad (4)$$

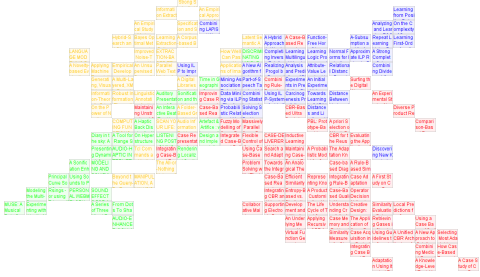
Assuming $\delta(E_i, E_j) = \hat{d}(E_i, E_j)$ also causes the stress measure to be sensitive to scaling: if we take a set of points A_1, simply scale their positions and call it A_2, stress will be found between the two sets. In visualization, scaling is an integral part of the process of displaying data on a given viewport. If we were to apply a "scale to fit" procedure, visualizing sets A_1 and A_2 would result in the exact same image, and yet they might have different stress measures when compared with a reference set.

As we understand that stress might not be a suitable metric to evaluate the placement strategy, due to reasons exposed by Paulovich et al. [26], we also used in our evaluation the Nearest Neighbors Precision metric (*nnp*), as defined by them. It computes the percentage of neighbors that belong to the same class as the reference element. It could thus be used to verify whether the resulting *incBoard* layout is consistent with assigned labels or classes attributed to data items. We considered the first 8 neighbors ($8 - nnp$), that corresponds to the immediate neighbors of a cell on the board.

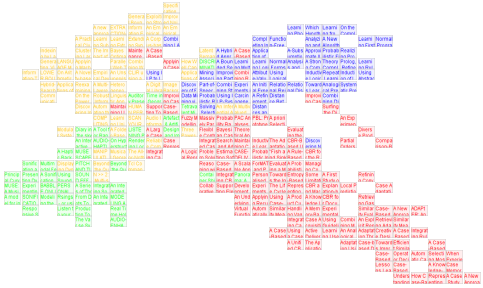
We recorded measures for complete layouts and also mean values for measures computed as elements are gradually added in a random order. These latter measures reflect the technique's performance while the layout is incrementally built. Whenever a technique is subject to variance, mean average and standard deviation (std.dev.) measures were computed over 10 distinct runs. Projections used for comparison with *incBoard* were built using the PEx Tool [27] and its default settings.

The first data set used consists of 675 scientific articles manually classified into one of four subject areas: Case-Based Reasoning (CBR), Inductive Logic Programming (ILP), Information Retrieval (IR) and Sonification (SON) – this same dataset, henceforth the CIIS corpus, has been used before to compare several multidimensional projection techniques [26]. Document content dissimilarity was measured using the cosine distance over a vector space model of the collection. Stop words were removed and frequency-based Luhn's cuts applied to select relevant terms.

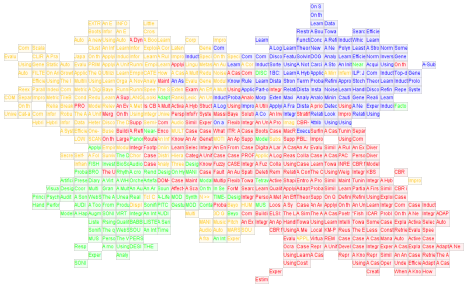
Figure 1 shows intermediate and final document maps of the corpus obtained with the proposed approach. Documents are colored according to their assigned classes. Notice that the general placement of classes remains globally stable as the visualization is constructed. Documents were added using the full mode until the 300^{th} element, from then on the insertion process switched over to the stochastic sampling mode. At that point, error calculations for each document used only the random and neighboring lists of elements. The size of the random and neighboring lists was set to 16 and 24 elements, respectively. The assigned classes played no role in the arrangement process, being used exclusively to select each element's display color.



(a) 178 documents



(b) 294 documents



(c) 675 documents

Figure 1: Three moments on the incremental construction of a visualization of the CIIS corpus. Color denotes manually assigned classes.

Table 2 shows stress and *nnp* results for the final layout and the compared techniques: (i) Projection by Clustering - ProjClus [25], (ii) Least Square Projection - LSP [26], (iii) Principal Component Analysis - PCA [12], and (iv) Sam-

mon's Mapping [30]. Stress used $\hat{d}(E_i, E_j)$ computed by the fitting algorithm proposed by Kruskal [15]. Results for both measures are very encouraging.

Table 2: Results of different projection techniques for the CIIS corpus. (best results in bold).

Technique	$8 - nnp$ ($\times 10^2$)	std.dev. ($\times 10^2$)	Stress ($\times 10^2$)	std.dev. ($\times 10^2$)
PROJ	67.8	8.8	**17.6**	0.7
incBoard(final map)	67.8	4.3	18.2	0.6
PCA	**85.0**	–	21.6	–
LSP	70.7	5.9	22.3	2.1
Sammon's	76.0	–	37.7	–

Table 3 shows results for stress and *nnp* computed while adding documents until reaching the total number of 675 documents. Their mean average is identified as *incremental*. Results are consistent throughout the process, indicating that views obtained at partial stages are also usable.

Table 3: Stress & *nnp* measures computed while documents are gradually added to a board of 675 CIIS documents using *incBoard*.

Documents on board	$8 - nnp$ ($\times 10^2$)	std.dev. ($\times 10^2$)	Stress ($\times 10^2$)	std.dev. ($\times 10^2$)
100	55.6	5.3	18.8	1.5
200	64.7	5.6	18.9	1.4
300	70.7	5.5	19.1	1.2
400	67.7	5.7	18.7	0.8
500	67.7	6.0	18.5	0.6
600	67.8	5.2	18.4	0.6
675	67.8	4.3	18.2	0.5
incremental	66	7.1	18.7	1.0

We noticed that maps with 200 elements or less consistently presented poorer results for the $8 - nnp$ measure, which could be evidence of a bias of the measure towards the size of the set (see Tables 3 and 4). For example, in a square with 16 elements belonging to two classes (8 elements on each), the lowest possible number of elements with a neighbor of a different class is 8 ($2 \times \sqrt{n}$), while in a square with 100 elements this number is only 20 ($2 \times \sqrt{n}$). In other words, the measure resembles the relation between perimeter and area of a shape, which is not constant for different areas. Based on this rationale, we should only compare *nnp* results on maps with roughly the same number of elements, and may also expect slightly lower results for the incremental averages in Tables 3 and 5.

The same corpus was also used in an example that resembles more closely the visualization of a dynamic corpus. In this second scenario, documents were incrementally added until 300 documents were placed on the board. After that, replacement takes in, with a document being removed whenever a new document is added, until all 675 documents in the corpus have been displayed. Figure 2 shows three distinct moments of the process. Indeed, inspecting the visualizations created along the process one observes that the general disposition of classes in the board is kept throughout.

Again, quantitative results are very consistent(see Table 4), highlighting the suitability of *incBoard* to display dynamic data sets.

The second case study was conducted on the well known

Table 4: Stress & *nnp* measures computed while documents are gradually replaced from a board with 300 CIIS documents using *incBoard*.

Documents added/ on board	$8 - nnp$ ($\times 10^2$)	std.dev. ($\times 10^2$)	Stress ($\times 10^2$)	std.dev. ($\times 10^2$)
100/100	56.92	4.56	18.56	1.09
200/200	65.58	3.12	19.11	1.19
300/300	69.86	2.71	19.44	1.35
500/300	64.42	4.65	18.91	0.66
675/300	68.45	4.84	18.51	0.71

Iris flower data [3], which has a total of 150 data items belonging to one of three flower species. This second data set was chosen to asses how our solution would perform when presented with a data set that is more easily presented in 2 dimensions. Euclidean distance was employed to compute similarity.

On this case study, stress results were fair (see Table 5), though not as good as in the previous one. From these results, one could argue that there exists a 2-dimensional solution for the projection of the Iris flower data set (stress for PCA is 0.1×10^{-2}). Therefore, the other techniques listed benefit from not being subject to the same constraints as *incBoard*, as they are free to position each data item at their ideal positions. Nonetheless, for more complex data sets, such ideal positions may not exist and, then, their advantage in this respect becomes irrelevant. Still, the *nnp* measure is again very close to the best results, possibly implying that even if an ideal positioning was not attainable, overall distribution of data items is good. Moreover, as it happened with the CIIS corpus, stress and *nnp* behaved consistently throughout the process (see incremental results in Table 5).

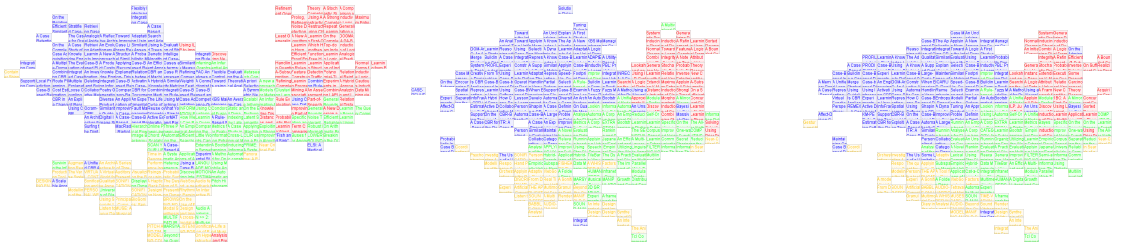
Table 5: Results of different projection techniques for the Iris flower data set. (best results in bold).

Technique	$8 - nnp$ ($\times 10^2$)	std.dev. ($\times 10^2$)	Stress ($\times 10^2$)	std.dev. ($\times 10^2$)
PCA	**93.6**	–	**0.1**	–
LSP	91.8	2.6	3.0	1.1
PROJ	83.6	6.3	10.2	1.7
incBoard(final map)	86.4	2.5	14.3	2.5
incBoard(incremental)	81.9	5.4	14.5	2.6

5.2 Proof-of-concept application

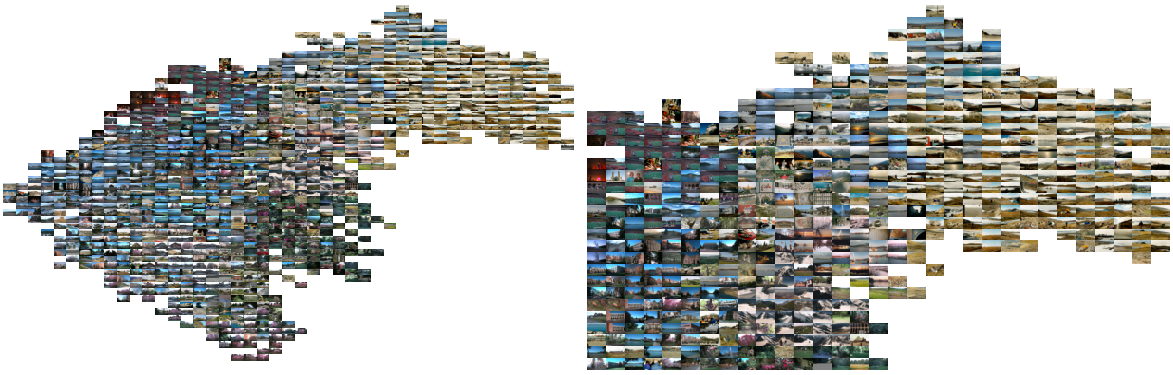
The next case study presents one extra application of our layout technique and visualization as a proof-of-concept. It lacks formal user evaluation, that should be performed once a more concrete task or usage scenario is defined.

It illustrates the applicability of our layout technique to display a collection of images, benefiting from both a layout that can arrange images by their content and that does not suffer from occlusion(See Figure 3). It uses a collection of 1109 manually tagged images from [20]. These tags describe objects and features found on the images. Dissimilarity was computed using the Jacquard coefficient to compare these manually assigned tag sets. Some clusters of images depicting the same object or place can be noticed in Figure 3(b). No image attribute was used in the comparison, and, yet, some areas where some colors are predominant can be found, perhaps reflecting some relation between tags and colors.



(a) initial set (b) intermediate set (c) final set

Figure 2: Three moments on a visualization of the CIIS corpus where only 300 documents are kept at a time. Color denotes manually assigned classes.



(a) (b)

Figure 3: Visualization of 1109 manually tagged images. Similarity determined by tag comparison. Whole map(a) and detail(b) shown.

6. CONCLUSIONS AND FURTHER WORK

We have built a visualization system that resembles the way users might layout constantly arriving elements on a board, based on their content similarity. On the core of our system is a placement approach that relies on the relative ranking of the displayed elements to position new elements on a partially filled board. Absolute positions or distances bear little meaning on the resulting visual space, nonetheless, its layout follows relative ranking relations as found on the original (or conceptual) space.

The system is particularly suitable to display dynamic data sets, as it is incrementally built and provides a built in mechanism for removing elements, while maintaining a compact layout and low computational cost. Moreover, adding a new element does not demand a complete re-arrangement of elements. An average of only 2.6%(std.dev. 0.1, computed over 10 runs) of existing elements were displaced after adding a new one using the CIIS corpus. This is an interesting property, as it allows users to track layout changes as element positioning gradually progresses.

Though the manually labeled document classes were not input into the layout process, classes were consistently kept in the same relative positions on the screen, during the gradual construction of the visualization of a whole corpus and also when elements of the corpus were gradually replaced. A consistent relative arrangement of classes was achieved even after a complete renewal of the viewed document set was forced, by replacing viewed documents until no document from the original viewed set remained. Although the *incBoard* approach assigns no explicit meaning to each visual dimension, this feature is particularly interesting, as it helps users to maintain a mental map of the arrangement.

The *incBoard* visualization does not suffer from occlusion of elements, and allocating a pre-defined screen space to each cell enables using sophisticated and highly informative glyphs. The user is also able to adjust, at any moment, the number of elements he or she wishes to display simultaneously. The approach could be easily integrated into an operating system to provide file system navigation, while using its standard icons and file representations.

The visualization is currently being improved to show similarity for each neighboring pair on the board. Hopefully, the new version will enforce identification of clusters and subsets of related elements by users.

A user evaluation of the suitability of the layout technique for specific tasks would be desirable to compare the incremental board space solution with other layout techniques. There is also room for improvement of specific steps of the process. For instance, the choice of the closest neighbor could use a better search strategy. Likewise, alternatives to the error measure adopted and which options to evaluate when choosing an element to move could be subject to further investigation. The current choices are the result of some early ad-hoc experimentation and follow the underlying reasoning presented here.

The next step on system development is to enrich the map with concepts and/or topic representations. Topics may be derived automatically from document contents using dynamically extracted Locally Weighted Rules [22]. These

topical markers, once placed on the board, could steer the placement of arriving elements. They could also be placed and positioned by users, who could then adjust the layout according to their specific interests and needs. Support to such operations is possible by adapting the incremental build operations. Another possibility is to allow direct manipulation of items on the board.

7. ACKNOWLEDGEMENTS

Roberto Pinho's research was conducted while visiting Drexel University under the supervision of Dr. Chaomei Chen and the support of a CAPES Grant. He wishes to thank Dr. Chen and fellow researchers at Drexel University for their invaluable contributions. The authors also acknowledge the financial support of FAPESP and CNPq (Grant 305861/2006-9).

8. REFERENCES

[1] D. Alahakoon, S. Halgamuge, and B. Srinivasan. Dynamic Self-Organizing Maps with Controlled Growth for Knowledge Discovery. *IEEE Trans. On Neural Networks*, 11(3):601–614, 2000.

[2] R. Amarasiri, D. Alahakoon, K. Smith, and M. Premaratne. HDGSOMr: A High Dimensional Growing Self-Organizing Map Using Randomness for Efficient Web and Text Mining. *Proc. IEEE/WIC/ACM Int. Conf. on Web Intelligence*, pages 215–221, 2005.

[3] A. Asuncion and D. Newman. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007.

[4] M. Barioni, E. Botelho, C. Faloutsos, H. Razente, A. Traina, and C. Júnior. Data Visualization in RDBMS. In *Proc. IASTED Int. Conf. Information Systems and Databases ISDB*, pages 264–269, 2002.

[5] W. Basalaj. *Proximity Visualization of Abstract Data*. PhD thesis, University of Cambridge Computer Lab, 2000.

[6] B. B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *UIST '01: Proc. of the 14th annual ACM symp. on User interface soft. and technology*, pages 71–80, New York, NY, USA, 2001.

[7] J. Blackmore and R. Miikkulainen. Visualizing high-dimensional structure with the incremental grid growing neural network. In *Proc. 12th Int. Conf. on Machine Learning*, pages 55–63, San Francisco, CA, USA, 1995.

[8] K. Borner, C. Chen, and K. Boyack. Visualizing Knowledge Domains. *Annual Review of Information Science and Technology ARIST*, 37:179–255, 2003.

[9] M. Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In *VIS '96: Proc. of the 7th Conf. on Visualization*, pages 127–132., Los Alamitos, CA, USA, 1996. IEEE Comp. Society. Press.

[10] C. Chen. *Information Visualization: Beyond the Horizon*. Springer, 2004.

[11] C. Chen. Citespace II: Detecting and visualizing emerging trends and transient patterns in scientific literature. *J. Am. Soc. Inf. Sci. Technol.*, 57(3):359–377, 2006.

[12] I. T. Jolliffe. *Principal component analysis [electronic resource]*. Springer, 2002.

[13] S. Kaski. *Data Exploration Using Self-Organizing Maps*. PhD thesis, Finnish Academy of Technology, 1997.

[14] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen. WEBSOM–Self-organizing maps of document collections. *Neurocomputing*, 21(1-3):101–117, 1998.

[15] J. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, 1964.

[16] J. Kruskal and M. Wish. *Multidimensional Scaling*. Sage Publications, 1978.

[17] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[18] M. Law and A. Jain. Incremental Nonlinear Dimensionality Reduction by Manifold Learning. *Pattern Analysis and Machine Intel., IEEE Trans. on*, 28(3):377–391, 2006.

[19] M. Law, N. Zhang, and A. Jain. Nonlinear Manifold Learning for Data Stream. *Proc. SIAM Data Mining*, pages 33–44, 2004.

[20] Y. Li, L. Shapiro, and J. Bilmes. A Generative/Discriminative Learning Algorithm for Image Classification. *Proc. 10th IEEE Int. Conf. on Computer Vision ICCV'05*, 2:1605–1612, 2005.

[21] A. Lopes, R. Minghim, V. Melo, and F. Paulovich. Mapping texts through dimensionality reduction and visualization techniques for interactive exploration of document collections. *Proc. of SPIE*, 6060:271–282, 2006.

[22] A. A. Lopes, R. Pinho, F. V. Paulovich, and R. Minghim. Visual text mining using association rules. *Comput. Graph.*, 31(3):316–326, 2007.

[23] A. Morrison and M. Chalmers. A pivot-based routine for improved parent-finding in hybrid mds. *Information Visualization*, 3(2):109–122, 2004.

[24] A. Nurnberger and M. Detyniecki. Visualizing changes in data collections using growing self-organizing maps. In *Proc. Int. Joint Conf. on Neural Networks IJCNN'02*, volume 2, pages 1912–1917, 2002.

[25] F. V. Paulovich and R. Minghim. Text map explorer: a tool to create and explore document maps. In *Proc. of the Conf. on Information Visualization IV '06*, pages 245–251, Washington, DC, USA, 2006. IEEE Comp. Society.

[26] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE Trans. on Visualization and Comp. Graphics*, 14(3):564–575, 2008.

[27] F. V. Paulovich, M. C. F. Oliveira, and R. Minghim. The projection explorer: A flexible tool for projection-based multidimensional visualization. In *Proc. XX Brazilian Symp. on Comp. Graphics and Image Proc. SIBGRAPI 2007*, pages 27–34, Washington, DC, USA, 2007. IEEE Comp. Society.

[28] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Evaluating a Visualization of Image Similarity as a Tool for Image Browsing. *Proc. of the 1999 IEEE Symp. on Information Visualization*, 1999.

[29] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Does organisation by similarity assist image browsing? *Proc. SIGCHI Conf. on Human factors in computing systems*, pages 190–197, 2001.

[30] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Trans. Comput.*, 18(5):401–409, 1969.

[31] J. A. Wise. The ecological approach to text visualization. *J. Am. Soc. Inf. Sci.*, 50(13):1224–1233, 1999.