Leveraging Layout with Dimensional Stacking and Pixelization to Facilitate Feature Discovery and Directed Queries

John T. Langton[1], David K. Wittenberg[2], and Timothy J. Hickey[2]

[1] Charles River Analytics Inc., 625 Mt. Auburn St., Cambridge, MA 02138 USA
{psyc@cs.brandeis.edu}
[2] Computer Science Department, Brandeis University
{tim, dkw@cs.brandeis.edu}

Abstract. Pixelization is the simple yet powerful technique of mapping each element of some data set to a pixel in a 2D image. There are 2 primary characteristics of pixels that can be leveraged to impart information: 1. their color and color-related attributes (hue, saturation, etc.) and 2. their arrangement in the image. We have found that applying a dimensional stacking layout to pixelization uniquely facilitates interactive data mining, directs user queries, and provides a type of visual principle components analysis. In this paper we describe our approach and how it is being used to analyze multidimensional, multivariate neuroscience data.

1 Introduction

There are many methods for multidimensional and/or multivariate visualization [1][3][4]. Some popular 2D approaches are scatter plot matrices [1] and parallel coordinates [5]. When faced with the task of visualizing a vast database of neuroscience data we chose to employ a form of pixelization because of the amount of information that can be displayed at one time [6]. This technique maps each data point of some set to a pixel in a 2D image, with the only constraint being a monitor's resolution. For our purposes, it meant that we could view an entire dataset of 1,679,616 single compartment neuron simulations in one display. This provided a global context for determining data trends and supporting exploratory analysis.

Keim and Ward have investigated the effects of different layout strategies for multidimensional visualization [15]. In the domain of pixelization, Keim has demonstrated the efficacy of pixel arrangements that are determined by the results of user queries [9] and has proposed algorithms for generating them [7]. In [10], LeBlanc et. al. introduced Dimensional Stacking which projects multiple dimensions onto 2 axes so that all are visible in one display. This technique is essentially a specialized layout of visual grid squares.

While LeBlanc et. al. make a subtle reference to the possibility of one data element being mapped to one pixel, the application of dimensional stacking as a layout scheme for pixelization has not been thoroughly investigated. We have found that combining both approaches has unique implications for facilitating data trend discovery. In particular, clustering within this scheme is entirely based on the order of dimensions on each axis. The task of interactively permuting dimension orderings can reveal functional dependencies on dimension values and direct user queries for further analysis.

We have applied these methods to the analysis of highly structured neuron simulation data where the independent and dependent variables are clearly delineated. In database terms, our dimensions are independent variables and together can be a considered a primary key. The rest of the columns are dependent variables or datapoint attributes and are determined by dimension values. As such, we cannot claim a global utility of our technique when applied to data that has a very different structure. However, these methods have proven quite useful for our purposes and we believe can be generalized further.

In this paper we detail our approach of applying a dimensionally stacked layout to pixelization. Section 2 provides a brief overview of pixelization, dimensional stacking, and related literature. Section 3 describes the generation of the data we were tasked to analyze, the motivation for our choice of visualization techniques, and the application of our approaches to that data. We detail the semantics of dimension ordering within the context of our approach in section 4. Both sections 3 and 4 use examples from our experience working with neuroscientists and analyzing multidimensional, multivariate neuroscience data. We specifically show how our methods interactively support feature discovery, data mining, and a type of visual principle components analysis. We conclude our findings in section 5 and provide recommendations for future work in section 6.

2 Background

2.1 Layout

In [15], Ward and Keim describe how layout can convey relations between graphical entities. They prescribe placing semantically similar items in close proximity to one another. This poses the issue of how to determine what constitutes semantic similarity. Grouping all red pixels together in a pixelization image may obscure a multimodal distribution of the data values associated with red pixels. Its usually important to retain dimensional context and impart not only attribute values but the dimension values that create them, all in the same visualization.

Keim presents two layout schemes for multidimensional pixelization, both of which map each dimension to a different window [9]. A user can view the same region in 2 or more windows, to decipher correlations and functional dependencies between dimensions. In what is termed a query-independent approach, the value of some attribute is mapped to pixel colors, the data is sorted according to that attribute, and then space filling curves are used to arrange the pixels in an image. In a query-dependent approach, an extra window is created with a layout that is driven by a user query. Keim develops a sophisticated metric for determining the distance of a data element from a user query. For the extra query-dependent window, the pixels for items that match the query are placed in the center of the display while the rest are distributed in a generalized spiral pattern, away from the center, in the order of increasing semantic distances. The technique for doing so is described in [7].

For our purposes, we needed to display both attribute values and the dimension values that created them, in the same image. This implicitly requires displaying all dimensions and their possible values in one image as well. We therefore could not use the layouts described above. To satisfy our requirements we applied a dimensional stacking layout to pixelization and represented query results as pixel attributes such as color.

2.2 Dimensional stacking

We devised a method of dimensional stacking based on the structure of our data [10] [16]. The result was a subtle variation on the original concept, specifically applied to pixelization. One can interpret our database or dataset as a function on tuples $T = (t_1, \ldots, t_n)$ where all of the t_i are small integers satisfying $t_i < D$ for some small number D. Each t_i is considered to be one dimension and the cardinality of possible values it can take on is its base. When all dimension bases are equal, T can then be viewed as a base D number with n digits.

For example, suppose $n = 4$ and the dimensions a, b, c, and d take on the values 0 and 1. We can interpret these dimensions as a 4 digit binary number. Ordering the dimensions in alphabetical order, the number 0101 would map to $a = 0$, $b = 1$, $c = 0$ and $d = 1$ or the decimal number 5. It is possible to have an odd number of dimensions and/or dimensions with unequal bases i.e. $0 <= t_i < j$, $0 <= t_{i+1} < k$ and $j \neq k$. We account for this below.

To get a 2D representation we map every combination of dimension values to a pixel coordinate. First we partition the dimensions into two groups. Though any number of dimensions can be in either group, we generally follow the rule: $X = n/2$ and $Y = n/2 + n \ mod \ 2$ where n is the number of dimensions. The decimal values of coordinates are calculated the same as in [10]. The reverse operation follows this algorithm:

1. Given some decimal number m and an order of dimensions, start with the right most (or least significant) dimension and iterate left
2. $t_n = m$ modulo (the base of t_n)
3. $m = m/$ (the base of t_n)
4. repeat for t_{n-i} until the value for t_0 is calculated

For the decimal value 7 and 3 dimensions with the bases 2, 3, 2:

1. $7 \ mod \ 2 = 1$ (keep for right most parameter = 1)
2. $7/2 = 3$ (ignore remainder, pass on as new num)
3. $3 \ mod \ 3 = 0$ (prepend to current answer = 01)
4. $3/3 = 1$(ignore remainder, pass on as new num)

5. 1 mod 2 = 1 (prepend to current answer = 101)
6. answer = 101

The x coordinate of each pixel is derived from X and y from Y. The width of a pixelization is the product of the bases in X, and the height is the product of the bases in Y. Our implementations support zoom and panning so we do not require a square image nor the blank rows for balance as in [10]. Instead of defining dimensions as "faster" or "slower", we extend the traditional terms of least and most significant digits to least and most significant dimensions.

2.3 Pixel attributes

In pixelization, the color of each pixel is generally determined by some value of the data element corresponding to that pixel. Different color attributes such as hue and saturation can be mapped to different values. If a discrete classification of data items exists then one can associate a distinct color to each class. A data attribute with continuous values can be normalized to lie in the range [0,1] and then mapped into a spectrum of colors. One can combine both discrete and continuous values by simply mapping each to a different color attribute i.e. hue for continuous values and color for discrete values. There are well-studied strategies for selecting effecitve color maps [14].

Keim shows how his distance measure can be mapped to pixel color intensity in addition to layout [8]. Distance, again, is defined as the relevance of a data item to a user query. We chose to allow user queries to directly define the color map. In our approach queries can return either a discrete or continuous value as described above. In the simple case, a query might test a data item's membership to some classification. In a more complicated case the user may compose a query that returns the result of some function on a particular data attribute.

2.4 Dimension ordering, "optimal" orders

Graphically clustering displays to show a distinct structure in data is often considered desirable [11][15][17]. In [2], Ankerst proposes clustering dimensions with similar attribute values in order to ensure their co-located arrangement in a pixelization. They further prove that this is an NP-complete problem in the context of certain layout schemes such as space filling curves and Keim's recursive pattern [7]. The computational complexity of finding optimal dimension orders for visual clustering depends upon the layout mechanism being used.

Dimension order has a particular significance for dimensional stacking [11][16]. Since every ordering of parameters corresponds to a distinct image, the orderings for n dimensions yield $n!$ images. If there are an even number of dimensions then half of these images are merely the result of swapping the x and y coordinates of each pixel. For an even number of dimensions, a brute force search for an optimal dimension order would require scoring $n!/2$ images. If there are an odd number of dimensions one must score all $n!$ images.

In our experience, we have found that there are often many different projections that yield substantive insights into the structure of the data. We discuss some automatic approaches to finding interesting projections below, but our experience has been that it is relatively easy to traverse the space of projections by swapping a pair of dimensions at a time and rapidly finding interesting, information-rich projections.

3 Data

3.1 Motivation for pixelization

The impetus for our use of dimensional stacking was borne out of a collaborative effort with colleagues in the Neuroscience department at Brandeis University. In a research effort by Prinz et. al., a large database of model neuron data was generated to better understand how individual membrane currents contribute to the electrical activity of a neuron [12]. The database was generated in a brute force style by several model neuron simulators running in parallel. Each model neuron was characterized by a sequence of 8 conductance parameters: KCa, Na, CaS, H, CaT, Kd, A, and leak. All of these could take on 6 different values and no two models shared the same combination of values. Simulations were run for every combination of parameter values and various attribute values recorded for each i.e. whether the neuron was silent, spiking, bursting, irregular. This yielded 6^8 model neuron simulations, each of which correspond to one row in the database tables. The primary key or model number for each of these tables is the model neuron conductance parameter values in the form of an 8 digit base 6 number. The 3GB database contained 2KB of data for each of the 1,679,616 model neurons.

An initial statistical analysis of the database revealed a bimodal distribution of spiking neurons dependent on period length. Some questions arose about the relationship between the parameter values of those neurons and their classification as fast/slow spikers, e.g. what are the locations of these two classes of spikers in conductance space (conductance space refers to the 8 dimensional space of all possible conductance parameter values)? How are they distributed throughout conductance space? What can be said about the border region between these distributions, if there is one?

The need for a visualization tool was evident. Such a tool would help not only with the questions at hand but could facilitate feature discovery. We evaluated some tools but found most were not suited for the amount of data we were working with. XGobi, for instance, would freeze before completely loading the data [13]. A requirements analysis was performed for a visualization technique that would be adequately expressive. Two characteristics of the database containing neuron information immediately eliminated the vast majority of approaches:

1. It was vast (1,679,616 rows per table).
2. It was multi-dimensional.

Our tool needed to represent a mapping from the 8 conductance parameters for each neuron model to its other attributes held in the database (e.g. activity type and details of the voltage trace). The desire was to display some attribute value for every model neuron in the same image. This would reveal the location of attribute values in the 8 dimensional conductance space and allow easy indexing to the parameter values that created them. It was a useful coincidence that the size of our dataset (1,679,616 points) was similar to the number of pixels on a modern computer screen. This suggested using one pixel per model neuron.

The database was generated before we started our work, so the decisions about how many parameters were of interest and how to discretize their possible values were made for us. The structure of the data led us to employ a version of dimensional stacking. In the next section we describe the application of pixelization with a dimensional stacking layout to our database.

3.2 Applying dimensional stacking and pixelization

During the process of applying dimensional stacking and pixelization to the neuroscience database, several implementations were used. A first pass was done in JScheme, a language for rapidly scripting Java GUIs paired with the power and syntactic simplicity of Scheme. Additional implementations used Matlab and Java. In the following sections we present a high level description of the features provided by our tools though implementation specifics are out of the scope of this paper.

Pixel layout Each model neuron was uniquely defined by its conductance parameter values, an 8-tuple of integers in the range $[0, 5]$. The conductances were KCa, Na, CaS, H, CaT, Kd, A, and leak and served as the dimensions of our data set. The entire space of possible parameter values and model neurons was therefore 8 dimensional. We reduced this to two dimensions by partitioning the parameters into two sets of four elements each, and viewing them as a pair of for digit base 6 numbers. These numbers served as the decimal coordinates of the pixel associated with the model neuron bearing those parameter values. More precisely, we chose a set of four conductances (independent variables) x_1, x_2, x_3, x_4 and computed a pixel's x coordinate as:

$x = x_1 * 6^3 + x_2 * 6^2 + x_3 * 6^1 + x_4 * 6^0$

The y coordinate was calculated in the same fashion using the remaining conductances. The concise nature of this equation is due to the fact that each dimension has a base of 6. The algorithm for deriving a decimal value from dimensions with differing bases (or possible value cardinalities) is described in [10]. Because pixel coordinates map directly to dimension values we were able to interactively report the values for each neuron model/pixel during mouse over events.

Fig. 1 illustrates four classes of neurons. Those with tonically spiking voltage plots are in red. The blue, green, and purple colors indicate neurons with bursting voltage plots where the number of spikes per second is in the range [0,10], (10,18], or (18,1000] respectively. The white pixels are either silent or irregular bursters; we are not considering those classes in this sequence of images.

The axes are labeled with the conductances assigned to each. The most significant dimensions are KCa and CaT. Fig. 3 is a clip from the lower left corner where the values for both are 0. Fig. 4 is another clip from the lower left corner where the values for Na and Kd are 0. Thus, if you break the square down into a 6X6 grid, all pixels in one of the largest square would have one value for KCa and CaT. You can then break that square down into a 6X6 grid and the largest squares there would be values for Na and Kd.

Pixel coloring 6^8 model neurons mapped to 6^8 pixels. 8 dimensions partitioned into two groups of four created a 1296 by 1296 square image. The colors assigned to pixels were directly specified by user queries written in SQL. Solid regions of color or repeating patterns represented data trends.

4 Dimension ordering

The ordering of dimensions (conductance values for our data) has a profound effect on the image resulting from our dimensional stacking method. For instance, the data and color assignments in Figs. 1 and 2 are identical. The only difference is the order of dimensions. One is not necessarily better than the other, but each reveals different information about the underlying data. For instance, in Fig. 2 it is actually quite apparent that when H and leak levels are 0, neurons usually do not have a bursting activity (there are very few blue pixels). This is not readily apparent in Fig. 1. Similarly, the features of Fig. 1 (e.g. predominance of spikers when $KCa < 1$) are not readily apparent in Fig. 2.

Because we had 8 dimensions, there were $8!/2 = 20{,}160$ possible projections or images. The division by 2 results from extracting identity rotations, i.e. swapping each pixel's x and y coordinate. This dynamic occurs in any dimension set with an even cardinality.

4.1 Semantic order effect

Fig. 1 clearly indicates that there is some structure in these classes of neurons. In particular, the fact that the red color predominates in the left and bottom edges of the image indicates that the tonically spiking neurons only occur when KCa or CaT are relatively low (e.g. less than 4) and that most spiking occurs when at least one of these values is less than 2. This is a feature of the 1st level dimensions in this particular projection.

Another feature clearly visible in this plot is that when KCa and Cat are both relatively large (at least 2), then the burst rate is relatively slow (less than 10) for all neurons with small values of Na or Kd. This is a feature of the "2nd level" of dimensional stacking.

A feature of the third level of dimensional stacking can be seen in the location of the fast bursters. It is clear, visually, that these purple pixels appear primaily in the lower right parts of the 3rd level grid squares when KCa and CaT are large (say at least 3). This is the condition that $A < CaS$.

In general, the most significant dimensions determine the clustering of an image. For instance, because KCa is the most significant dimension on the X axis, and CaT is the most significant on the y axis, the red pixels (which represent spiking neurons) cluster when their values are near zero. we can deduce that neurons tend to spike when there are low concentrations of KCa or CaT.

In this way, one can perform a type of interactive principle components analysis by reordering dimensions to maximize clustering of a particular phenomenon or pattern. One visual analysis technique is to alternate between finding principle components through dimension reordering, and using the resulting visualizations to inform queries. This results in an organized approach to feature discovery and directed queries, allowing users to mine the data using hypotheses and discover unforeseen patterns. For instance, in Fig. 1 there are white grid points throughout the image wherever H = 0. We could reposition H as the most significant dimension of 1 axis to try and cluster this phenomenon and determine the effects of this parameter. This is done in Fig. 2 which shows indeed that there is very little spiking and almost no bursting when H and leak are less than 1.

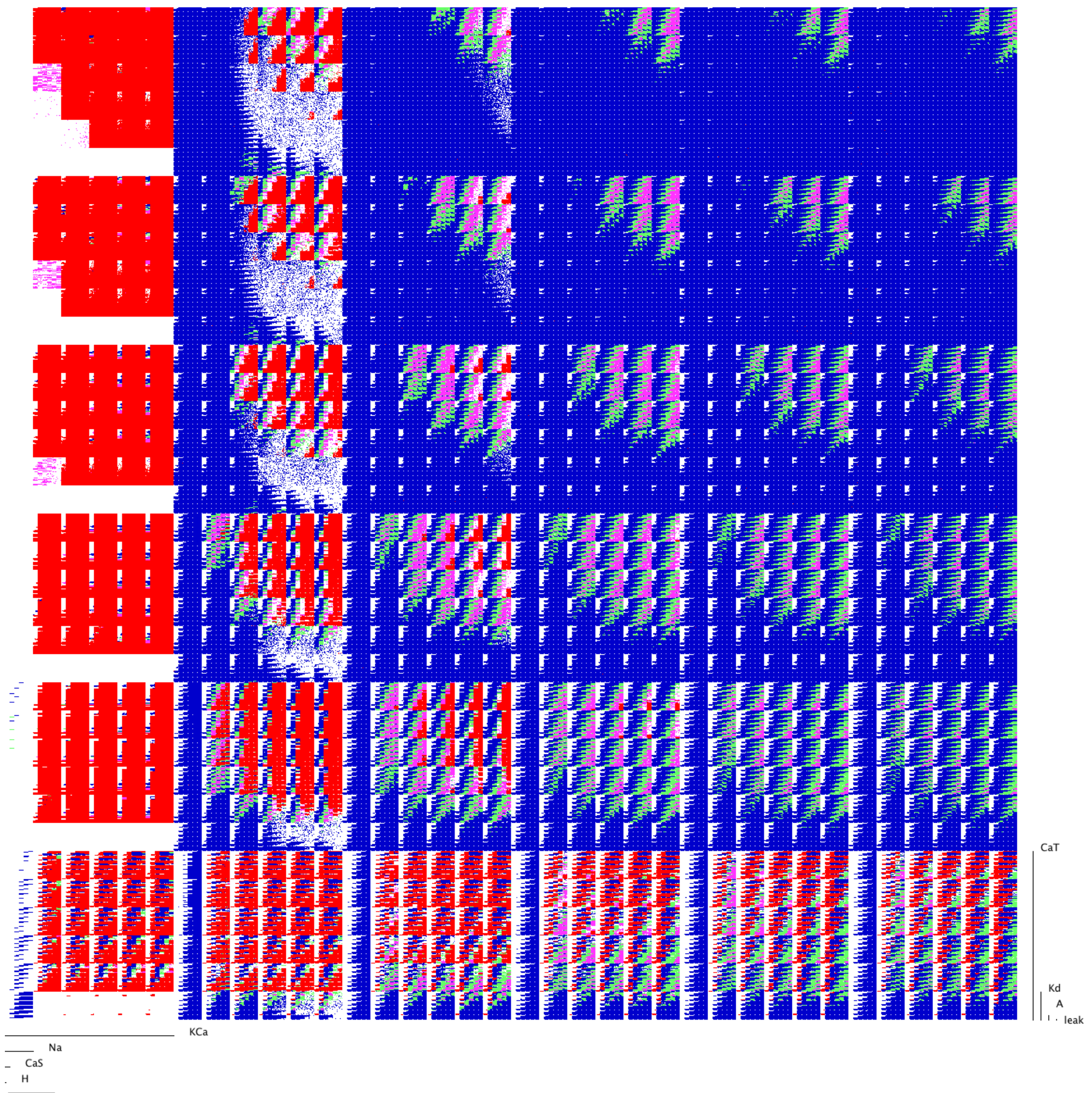
The dimensionally stacked pixelization of neuroscience data provided a number of insights. Over a period of minutes or hours the conductances of a real neuron may change incrementally. With our layout, the neuroscientists would be able to trace the path of a single neuron through the conductance space as its activity changed. We believe similar analysis could be performed with other data sets.

4.2 Clustering and "optimal" orders

As mentioned above, in many cases there is no single optimal order that provides maximal information from a dataset. Nevertheless, there has been substantial research in finding automatic algorithms to find ordering that are potentially information rich.

Yang presents an approach termed DOFSA, an automatic and interactive way of ordering, spacing, and filtering dimensions based on similarity [17]. In their algorithm, dimensions with similar value distributions are clustered and organized into a hierarchical tree structure. Ordering dimensions is then performed as a depth first traversal of the tree. This changes the search problem for finding an optimal dimension order to a sorting problem thereby reducing the computational complexity.

In dimensional stacking, dimension similarity is not necessarily relevant to visual clustering or an optimal dimension order. In the simple case, a data trend may be determined by only 1 dimension. In our visualization the trend would only be apparent when that dimension is the most significant one on the x or y axis. Grouping according to similarity would not accomplish this. In general, techniques that cluster according to dimension similarity inform us of what dimensions could be grouped together, but do not prescribe a global order, i.e.



🟥 spikers

🟦 bursters with < 10 max per second

🟩 bursters wtih > 10 and < 18 max per second

🟪 bursters with > 18 max per second

Fig. 1. A view of number of maxima per second.

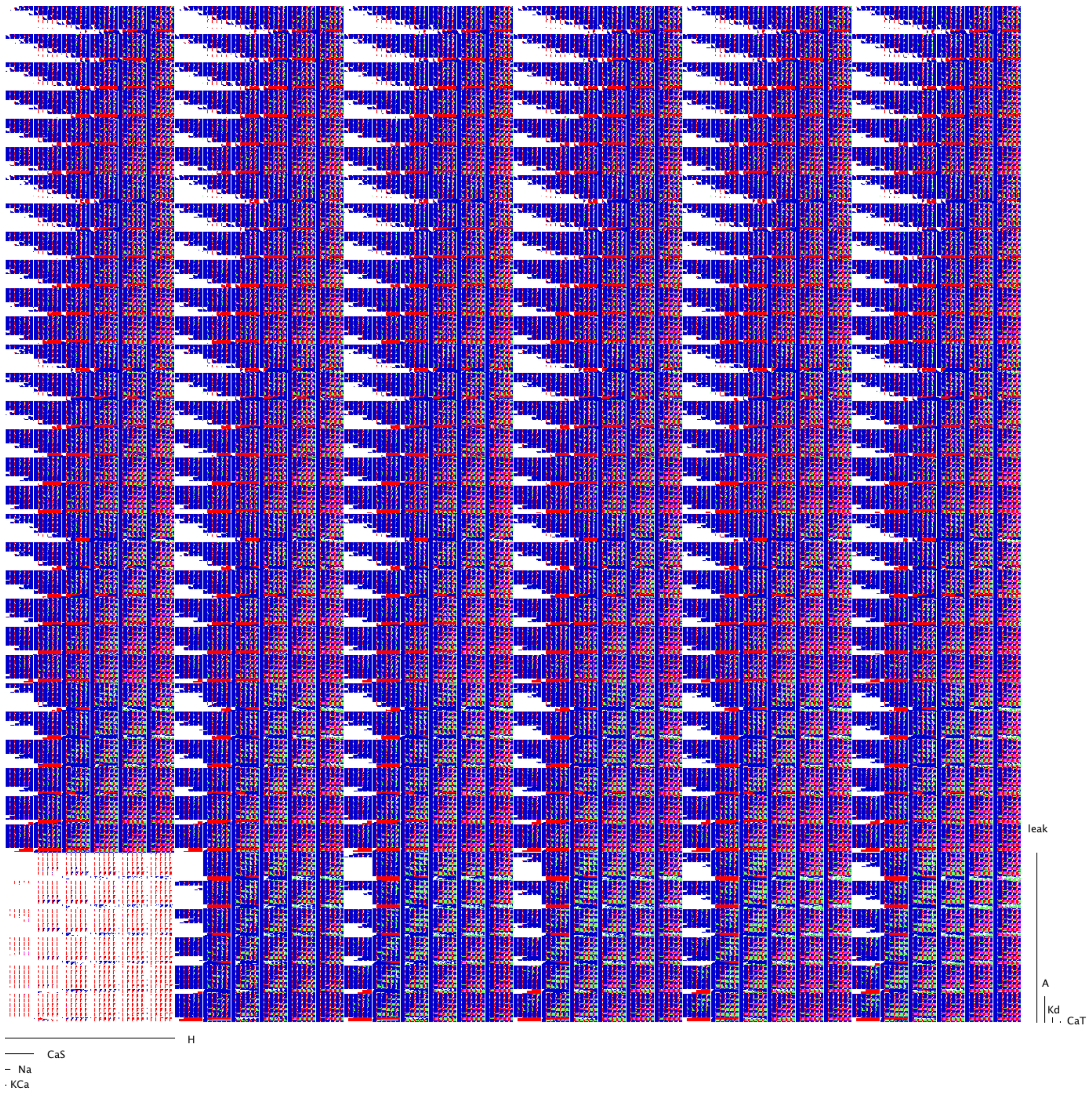


Fig. 2. A view of number of maxima per second (using a less informative projection than in Fig. 1).

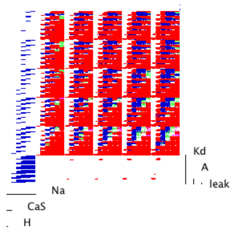


Fig. 3. A closeup of the lower left corner Fig. 1 where KCa and Cat are less than 1.

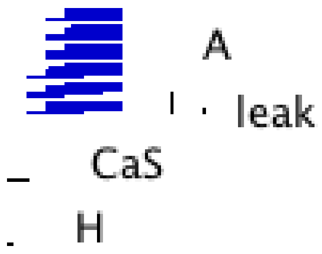


Fig. 4. A closeup of the lower left corner of Fig. 3 where Kd and Na are also less than 1.

what dimension goes first. Grouping all pixels of a certain color may also occlude certain facts such as a multimodal distribution. We may wish to see 2 distinct clusters, one determined by dimension A and one by dimension B.

Peng defines a measure for the clutter in a dimensional stacked image as the “number of isolated filled bins” / “number of total occupied bins” [11]. For our use of pixelization with dimensional stacking, simply replace “bins” with “pixels.” Determining whether a single pixel is connected on any side with another pixel of similar color can be quite expensive over many millions of iterations. Our neuroscience colleague Dr. Adam Taylor proposed a simpler and computationally feasible measure of clutter which counts the number of color changes in each row and column of pixels.

5 Conclusions

Applying a multidimensional layout to pixelization provides great utility for exploratory analysis of multidimensional data. Users can directly query data to color pixels or interactively reorder dimensions to find patterns and determine principle component/dimension effects. Both of these approaches can be combined in an iterative process where found features inform user queries.

Displaying all dimensions in one image is highly informative as users can see the dimension values that produce the attribute values they are viewing. With a dimensional stacking layout, pixilated data, query specified color maps, and interactive dimension ordering, large databases can be interactively explored in an information rich visual environment.

6 Future work

One issue is the fact that dimensional stacking works best with a small set of monotonically increasing values allowed for each dimension (and the simplest case is when all dimensions have the same number of values). This fact is noted by LeBlanc, Ward, and Keim. Our neuroscience data fit perfectly into this paradigm however other data sets may not. One question is how our approach could be applied to other domains. Some methodology for determining dimensions vs. attribute values and discretizing the latter would be necessary. One possible area of research is to investigate the general applicability of our approach and construct such a methodology. Also, much of the background literature on clustering and clutter reduction describe a very small set of heuristic algorithms. One possible area of investigation is to employ methods such as simulated annealing, genetic algorithms, or branch and bound to find information rich dimension orderings.

7 Acknowledgements

We wish to thank Dr. Astrid Prinz for allowing us to use her neuron simulation data, and Dr. Eve Marder for introducting us to this problem, and Dr. Adam Taylor for showing us the optimization algorithm. Our work on visualization techniques has been propelled forward by many helpful conversations with these neuroscientists.

References

1. D. F. Andrews. Plots of high dimensional data. *Biometrics*, 28:125–136, 1972.
2. M. Ankerst, S. Berchtold, and D. A. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In *INFOVIS '98: Proceedings of the 1998 IEEE Symposium on Information Visualization*, page 52, Washington, DC, USA, 1998. IEEE Computer Society.
3. H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68:361–368, 1973.
4. S. K. Feiner and C. Beshers. Worlds within worlds: metaphors for exploring n-dimensional virtual worlds. In *UIST '90: Proceedings of the 3rd annual ACM SIGGRAPH symposium on User interface software and technology*, pages 76–83, New York, NY, USA, 1990. ACM Press.
5. A. Inselberg and B. Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *VIS '90: Proceedings of the 1st conference on Visualization '90*, pages 361–378, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

6. D. A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, 2000.
7. D. A. Keim, M. Ankerst, and H.-P. Kriegel. Recursive pattern: A technique for visualizing very large amounts of data. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 279, Washington, DC, USA, 1995. IEEE Computer Society.
8. D. A. Keim and H.-P. Kriegel. Visdb: a system for visualizing large databases. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, page 482, New York, NY, USA, 1995. ACM Press.
9. D. A. Keim and H.-P. Kriegel. Visualization techniques for mining large databases: A comparison. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):923–938, 1996.
10. J. LeBlanc, M. O. Ward, and N. Wittels. Exploring N-dimensional databases. In A. Kaufman, editor, *IEEE Visualization: Proceedings of the 1st conference on Visualization '90*, pages 230–237. IEEE Computer Society Press, 1990.
11. W. Peng, M. O. Ward, and E. A. Rundensteiner. Clutter reduction in multi-dimensional data visualization using dimension reordering. In *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*, pages 89–96, Washington, DC, USA, 2004. IEEE Computer Society.
12. A. A. Prinz, C. P. Billimoria, and E. Marder. An alternative to hand-tuning conductance-based models: Construction and analysis of data bases of model neurons. *Journal of Neurophysiology*, 90:3998–4015, Dec 2003.
13. D. F. Swayne, D. Cook, and A. Buja. Xgobi: Interactive dynamic data visualization in the x window system. *Journal of Computational and Graphical Statistics*, 7(1):113–130, 1998.
14. E. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, 1983.
15. M. Ward and D. Keim. Screen layout methods for multidimensional visualization, 1997.
16. M. O. Ward. Xmdvtool: integrating multiple methods for visualizing multivariate data. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 326–333, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.
17. J. Yang, W. Peng, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. In *IEEE Symposium on Information Visualization*, page 14, 2003.