Improving the Fidelity of Contextual Data Layouts Using a Generalized Barycentric Coordinates Framework

Shenghui Cheng and Klaus Mueller

Visual Analytics and Imaging Lab, Computer Science Department, Stony Brook University and SUNY Korea

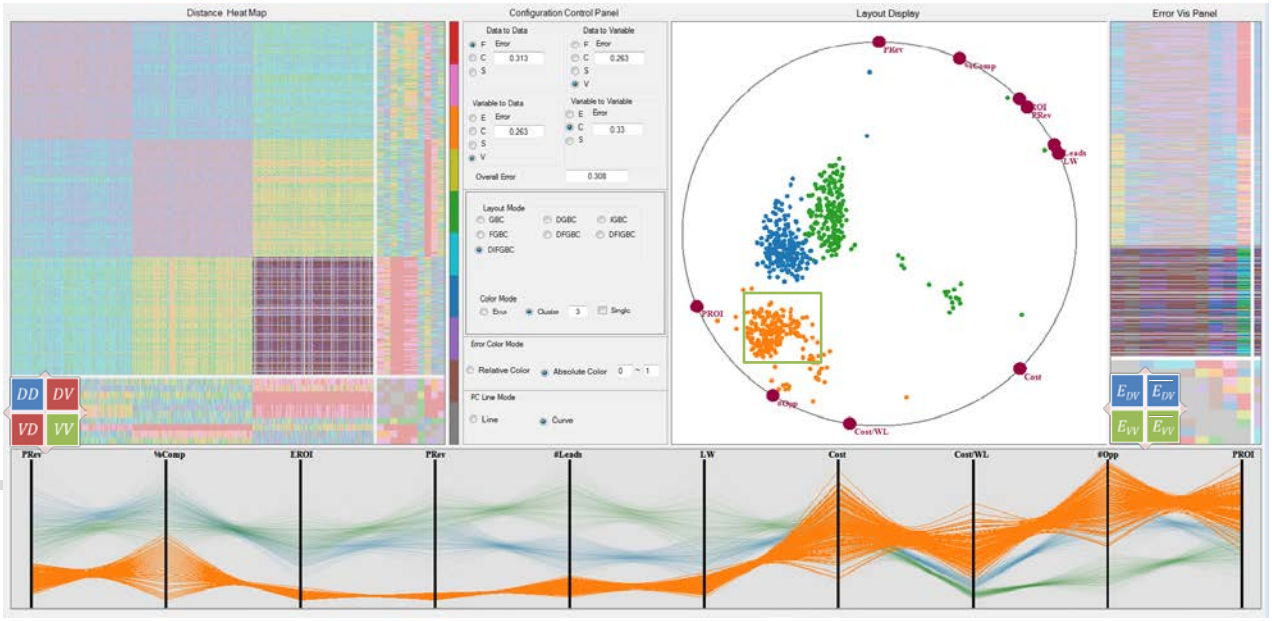

Fig. 1 The interface of our system demonstrating the linked display functionality – the highlighted parts are linked with the chosen area.

ABSTRACT

Contextual layouts preserve the context of the data with the associated attributes (variables). However, their linear mapping causes errors in the layout – similar data points and variable nodes may not map to similar regions, and vice versa. In this paper, we first unify the various data layout schemes and choose the Generalized Barycentric Coordinates (GBC) plot as the standard way to describe them. Second, we propose three algorithms – distance spaced layout, iterative error reduction, and force directed adjustment – to reduce the layout error of variables to variables, data to variables and data to data, respectively. We find that the combination of these three algorithms can yield large improvements in the layout error and so achieve a more comprehensive layout. Third, we describe an interface, the GBC Error Explorer, which allows users to explore the error using a variety of visualization schemes combined with some interactions.

Keywords: Visual analytics, generalized barycentric coordinates, multivariate data, contextual layout.

Index Terms: H.5.2 [Information Interfaces and Presentation]: User Interfaces - Graphical user interfaces (GUI), I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction Techniques

Email:{shecheng, mueller}@cs.stonybrook.edu

1 INTRODUCTION

Numerous methods have been described for the data matrix visualization. Methods that solely support the identification of clusters and their outliers, such as multidimensional scaling (MDS) [11] or t-SNE [8] are typically oblivious to the attributes of data. But there are settings in which it can be of interest to see the data points in relation to their attributes. For example, an investor might want to see companies in context of the value metrics of their stocks, such as earnings per share, price to earnings ratio, etc. This investor would pick those stocks that best fit his strategies. Such operations are not supported by MDS or t-SNE.

Visualization can be a good medium to first assess the overall data, here the stock market, and then focus on the market segment of interest – a class of stocks with the certain desirable constellation of metrics. There are multiple ways to achieve this. In the method of parallel coordinates [6], the attributes define the vertical axes and the data points form piecewise linear lines going across these axes, called *polylines*. The investor would then filter the stocks along his or her most salient metrics and so isolate the most desirable stocks.

Another method lays out the data points in the context of the attributes, and we shall refer to them as *contextual data layouts*. In these methods, the attributes form special nodes on the data canvas where data points that are 'stronger' in certain attributes also come to rest more closely to these attributes (although there can be significant errors – see below). Examples of these types of visualizations are RadViz [14], Star Coordinates [10], and Gravi++ [15]. In this case, the investor would focus on the attribute nodes of greater interest and look at the data points in their neighborhoods. The investor would also be able to assess and recognize conflicts in his set of criteria. There might be no stocks that can fulfill two competing criteria and so he or she would have to make certain trade-offs.

While the contextual layouts are convenient in that they do not require much filtering, they require other types of interaction, mostly to reduce the errors that result from the attracter/deflector spring-like layout schemes. Often data points that are not related at all come to rest very closely to one another, and moving the attribute points in an interactive fashion can reduce, but not completely eliminate this error, at least not in general.

Our work focuses on contextual layout displays and the errors they commit. We find they are all special forms of the GBC plot [12][13] - the attributes constitute the vertices of a *D*-sided polygon (where *D* is the number of attributes) and the data points are placed in its interior. However, even the GBC plot does not preserve accurate relations between the data and the attribute points, and to address this problem we describe a set of practical algorithms which automatically adjust the locations of both data and attribute points such that these relations are better preserved. Finally, we provide an interface that allows users to explore the various layout errors using different visualization schemes augmented with interactions.

Our paper is structured as follows. Section 2 presents related work. Section 3 provides theoretical aspects. Sections 4 and 5 describe the GBC plots and our various layout improvements. Section 6 presents an evaluation. Section 7 describes our diagnostic interface for parameter tuning. Section 8 ends with conclusions.

2 Related Work

The visualization of high-dimensional datasets essentially follows three major paradigms – parallel coordinates, scatterplots, and 2D space embeddings. Since the visualization of high-dimensional data on a 2D canvas is inherently an ill-posed problem, there is no method without drawbacks. Parallel coordinates, and its radial version, the star plot [2], have the least ambiguity in the 2D mapping process and the serialization of the high dimensional space into the parallel axis configuration allows all attributes to be seen at once. However, the overplotting of polylines can become a significant problem once the number of data points grows moderately large.

Scatterplots suffer less from overplotting, but the projection operation can lead to ambiguities as points located far away in high-dimensional space may project to similar 2D locations. Assembling all possible axis-aligned scatterplots into a scatterplot matrix [4] or supporting the projections by an interactive view manipulation system [12] can help but both require effort to navigate. Similar to the star plot, the method of star coordinates [10] arranges the attribute axes in a radial fashion but instead of constructing polylines it plots the data points as a vector sum of the individual axis coordinates. However the locations of the data points are not unique and so an interactive interface is provided that allows users to manually rotate and scale axes to resolve ambiguities, at least partially.

Many of the ambiguity problems can be overcome by embedding the high-dimensional space onto a 2D canvas via an optimization strategy (MDS, t-SNE, etc.) which seeks to preserve the high dimensional distances – or the statistics – of all point-pairs in the 2D layout. In this way the viewer can easily appreciate neighborhood relations and obtain a good overview of the space quickly. However, as mentioned, this method also has shortcomings – the mapped data points no longer maintain any context with the attributes as this information is typically not preserved in the non-linear mapping. We also use optimization for the 2D layout but retain this context.

Our method generalizes systems that arrange the nodes representing the attributes along a convex shape and lay out the data points in the interior of this shape. RadViz [3] uniformly spaces the attributes as *dimensional anchors* along the circumference of a circle. The location of the data points is then determined by a weighting formula where data point attributes with higher values receive a higher weight and so increase the attraction of the point to the corresponding anchor points. However, similar to star coordinates, this can lead to location ambiguities which can be reduced by reordering the anchor points manually or algorithmically. Gravi++ [14] uses a different weighting formula but also spaces the attributes at uniform distances onto an encompassing circle. In GBC[13] the enclosing primitive is a general convex polygon for the visualization [12]. Finally, even more general is the Dust & Magnet system [16] which allows one to move and adjust the weights of the attributes. None of these methods can guarantee nearby data and variables points are actually neighbors in high-dimensional space.

3 Theoretical Considerations

Let DM be the *data matrix* with m rows and n columns,

$$DM = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix}$$

where the rows denote the data points, the columns denote the attributes and x_{ij} is the data value in the ith row and jth column. Without loss of generality, we assume DM is normalized to [0, 1]. Let *D* be the *data points* (we shall simply refer to them as *data*):

$$D_i = [x_{i1}, x_{i2}, \dots, x_{in}] \qquad (i = 1,2, \dots, m)$$

Let *V* be the data attributes (we shall refer to them as *variables*).

$$V_j = [x_{1j}, x_{2j}, \dots, x_{mj}]^T \qquad (j = 1,2, \dots, n)$$

where *T* is the transpose function.

The methods we discussed above map data and variables into 2D layout space. We denote P and v as their locations respectively.

3.1 The Space of Contextual Layout Methods

Table 1. The features of different layout methods

Method	$VF(v_i)$	$MF(P_i)$
Radviz	$v_i = \left(r \cdot \cos\frac{i}{2\pi}, r \cdot \sin\frac{i}{2\pi}\right)$	$\sum_{j=1}^{n} \frac{x_{ij}}{\sum_{k=1}^{n} x_{ik}} v_j$
Star Coordinates	$v_i = \left(r \cdot \cos\frac{\theta_i}{2\pi}, r \cdot \sin\frac{\theta_i}{2\pi}\right)$ Or other	$\sum_{j=1}^{n} x_{ij} v_j$
Gravi++	$v_i = \left(r \cdot \cos\frac{\theta_i}{2\pi}, r \cdot \sin\frac{\theta_i}{2\pi}\right)$ Or other free layout	$\sum_{j=1}^{n} \frac{s_j x_{ij}}{\sum_{k=1}^{n} s_k x_{ik}} \cdot v_j$
Dust & Magnet	$v_i = \left(r \cdot \cos\frac{\theta_i}{2\pi}, r \cdot \sin\frac{\theta_i}{2\pi}\right)$ Or other free layout	$\sum_{j=1}^{n} a_{ij} x_{ij} \cdot v_j$
GBC	$v_i = \left(r \cdot \cos\frac{\theta_i}{2\pi}, r \cdot \sin\frac{\theta_i}{2\pi}\right)$ Or other convex polygon	$\sum_{j=1}^{n} \frac{x_{ij}}{\sum_{k=1}^{n} x_{ik}} v_j$
Remarks	$\theta_1 + \sum_{i=2}^{n}(\theta_j - \theta_{j-1}) = 2\pi$. s_j stands for the strength multiplicator of v_j. a_{ij} is the attraction between dust *i* and magnet *j*. r is the circle radius.	

We argued Radviz, Star Coordinates, Dust & Magnet, and Gravi++ are similar - they all arrange the variables as vertices in the outward periphery of the data points, providing context. To unify them into a common framework, we need a unified notation. We consider two factors: (1) the layout method for the variables vertices, VF, and (2) the data mapping function MF, shown for all methods in Table 1.

For VF, a circular layout is most common, so for this paper, we only consider this type of arrangement for the variables. The MF function, on the other hand, uses slightly different forms of weights to compute the variable node locations. The mapping concept is

identical – all apply a linear function – just some methods perform normalization and others do not.

While the GBC as described in [12] is more flexible in that it supports generalized polygons, we use it as the standard configuration – a polygon embedded into a circle – to describe the other layout algorithms. The GBC plot on an equilateral polygon is essentially Radviz. We begin with this plot and generalize to others.

4 The Generalized Barycentric Coordinates (GBC) Plot

The GBC plot is derived from GBC interpolation [12] which extends the method of barycentric interpolation from triangles to multi-vertex convex polygons. The task is to interpolate the value of an interior point P from the values stored at the polygon vertices v_i. Referring to Fig. 2, the interpolation weight w_i of v_i for P is:

$$w_i = \frac{\cot(\alpha)+\cot(\beta)}{\|P-v_i\|^2}$$

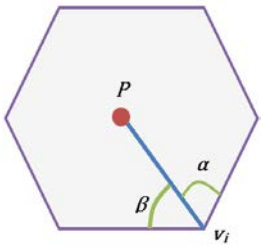

Fig. 2 The GBC Plot.

The interpolated value Pv at P is:

$$Pv = \sum_{i=1}^{n} w_i v_i \text{ where } a_i = w_i / \sum_{k=1}^{n} w_k \text{ and } \sum_{i=1}^{n} a_i = 1$$

The GBC plot uses GBC interpolation in reverse fashion. It seeks to compute the position of P in a convex polygon in which each vertex is assigned to one of the attributes. This replaces Pv by the 2D vector P, and the v_i by the 2D coordinates of the attribute vertices. We then set the weights to be the values of the n-dimensional vector, normalize them to compute the a_i, and finally use the 2D coordinates of the attribute vertices to interpolate the 2D coordinate of P.

4.1 The Distance Matrix

The GBC plot can show the distances of data to data, data to variable and variable to variable. The combination of distance matrices

$$C = \{DD, DV, VV\}$$

where DD, DV and VV store the pairwise distance of data points, data points to variables and variables respectively.

As mentioned, there are various measures which are suitable to express distance. We would like to choose the *Euclidean Distance* for *DD*. For *DV,* it is good to use the value at this dimension. However, we should use *1-vaule* since distance and value have opposite meaning. For *VV*, we would like to pick *1- correlation*. Let F be the set of *Distance Metrics*, then

$$F = \{Euclidean\ Distance\ |1 - vaule\ |1 - \ correlation\} \quad (1)$$

4.2 GBC Layout Error

The GBC plot can show the distances of data to data, data to variable and variable to variable, thus the error also combines these three types. We denote E as the error, where E_{DD}, E_{DV} and E_{VV} represent the error of data to data, data to variable and variable to variable respectively; E_A is the overall error of the GBC mapping. For more details about the definition, see Section 6. In addition, there is also the error resulting from the GBC layout itself. We will discuss how to deal with this kind of error in Section 7.

5 Our GBC Extensions for More Accurate Layouts

As discussed, the GBC plot has errors –E_{DD}, E_{DV} and E_{VV}. To reduce the error, we first analyse and reduce each type of error separately, and then combine these effects together to reduce E_A.

5.1 Test Datasets

We chose the following data sets to demonstrate our algorithms:

1. **Cars dataset** – 392 cars with 7 attributes.
2. **Sales campaign dataset** – 600 data items with 10 attributes.
3. **Bike dataset** –17,389 instances with 16 attributes.

The GBC layouts for these datasets are shown in Fig. 9 (a).

5.2 Distance Spaced GBC Plot Layout (DGBC)

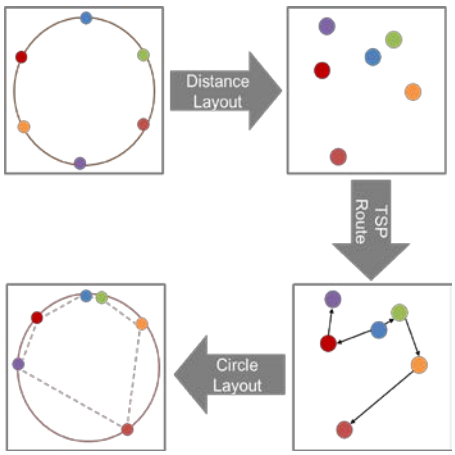

Fig. 3 Distance Spaced Layout Pipeline

Algorithm. 1 Distance Based Layout.

Input: The distance matrix (VV)
Output: The variables locations v

V=TSP(VV) //Reorder the variables. $V_{\mathcal{N}(i)}$ is the circle
sum VV=$\sum_{i=1}^{n} F(V_i, V_{\mathcal{N}(i)})$ // layout neighbour of V_i.
$angle_0 = 0$
for $i = 2{:}n$
    $angle_i = angle_{i-1} + 2\pi \frac{F(V_i, V_{\mathcal{N}(i)})}{sumVV}$
endfor
for $i = 1{:}n$ // Lay out the variables around the circle.
    $v_{i_x} = r \cdot cos(angle_i)$
    $v_{i_y} = r \cdot sin(angle_i)$
endfor

We begin with the E_{VV}. The variables are arranged around the circle – this type of layout maps the data from high dimension to one dimension. One way to achieve this is by projecting the distance matrix into 1D using MDS. However, we cannot guarantee this method provides a good ordering since MDS becomes increasingly error-prone as the distance matrix increases.

Another and more direct way to obtain a linear ordering of the vertices on the polygonal hull is by arranging the vertices through an approximate Traveling Salesman Problem (TSP) solver that operates on the matrix of pairwise correlation distances among the variables. It would choose the minimum length edge as the start edge and keep adding the nearest variables to the endpoints. TSP has been successfully employed to determine a good axis ordering for parallel coordinates [18]. The application for the current case is similar – it also uses the similarity of variables to provide an ordering, but now we also space them apart according to the similarity.

We place all attribute vertices on a circle, ordered by the distance-based TSP solver and spaced apart by the pairwise distances. The process is illustrated in Fig. 3 and the algorithm is given in Algorithm 1. Fig. 9 (second column) shows the outcome of this experiment for the three datasets we tested. We observe a much improved class separation for all of them. We also observe, from Table 2, that the error of variable to variable is reduced.

5.3 Iterative GBC Plot Error Reduction (IGBC)

Next we aim to reduce E_{DV}. In the GBC plot, a data point's value can be gauged by its location – if it is located close to a given variable point then it has a high value in the corresponding attribute, and vice versa. Hence, each variable point has a set of iso-contours where a data point's value is constant. In this paper, we restrict our study to linear contours, but an extension to non-linear contours would follow similar error-reduction principles.

Our method seeks to reconstruct an error polygon for each data point and iteratively reduces the size of this polygon. Fig. 4 provides an illustration and Algorithm 2 lists the pseudo code.

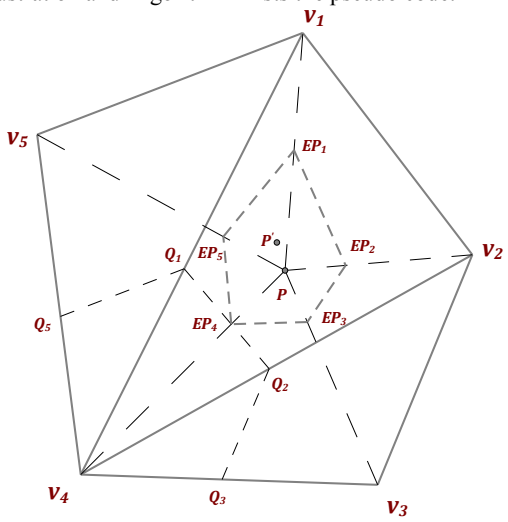

Fig. 4 The error polygon

The first assumption our algorithm makes is the existence of a set of distance contours that encode the importance of a variable to a given data point. Suppose we have the variables vertices v_1, v_2, v_3, v_4, v_5 and a test data item $(x_1, x_2, x_3, x_4, x_5)$ with its mapping location as P. Fig. 4 examines the distance contours for V_4. Assuming the data item has been normalized to a unit vector, the maximum importance a variable can have is 1.0. This would mean in the case examined that x_4=1.0 and so P would coincide with v_4 in the plot. In contrast, if x_4=0.0 which is the minimum importance, then with the current vertex ordering P would need to fall on the edge v_5v_1, v_1v_2 or v_2v_3. Any other value would lead to a placement of P onto some contour in between. Fig. 4 shows the contour $\overline{Q_5Q_1Q_2Q_3}$ for x_4=0.6. It is constructed by connecting v_4 with all vertices v_i and marking the points Q_i where $(\overline{v_4Q_i})/(\overline{v_4v_i}) = 1 - 0.6$. Connecting these points yields the contour.

Next we find v_4 on the error polygon (marked as EP_4) by intersecting the contour with the line that connects v_4 with P. Performing this procedure for all variables yields all vertices of the error polygon (marked as polygon $\overline{EP_1EP_2EP_3EP_4EP_5}$). The iterative step concludes by moving P into the center of the error polygon, marked as P', and then a new iteration begins.

In practice, we iterate about 20 times which completes in a couple of seconds and so does not cause a significant performance drop. The result of this algorithm is shown in Fig. 9 (third column). We observe the IGBC scheme also brings improvements in terms of cluster separation, but not as strong as DGBC.

The change of error is shown in Fig. 5. We observe that E_{DV} converges for all three test data sets. But we also see here (and in Table 2) this optimization scheme yields large improvements only for the car and bike data but not for the sales campaign data. This is because the campaign data is already well distributed (see Fig. 1).

5.4 Force Directed GBC Plot Adjustment (FGBC)

What remains is the E_{VV}. We can adjust the locations of the data points via traditional MDS to reduce this error. A popular MDS

Algorithm. 2. Iterative Error Reduction.

Input: ***DV***, *P*, *v*, error threshold, maximum iterations .
Output: the data points locations.

```
while E_DV < threshold || I_DV > max-threshold
  for each data point P
    for each variable vertex v_j
      Compute distance contour.
      Compute error polygon vertex EP_j.
    endfor
    Construct error polygon EP formed by all EP_j .
    Move P to the center of EP.
  endfor
  Compute E_DV and iterations I_DV.
endwhile
```

Algorithm. 3. Force Directed Adjustment

Input: ***DD***, *P*, *v*, error threshold, maximum iterations.
Output: the data points locations.

```
if E_DD < threshold || I_DD > max-threshold, return.
  for each data point D_i
    Compute the forces f_j according to the error.
    Compute the resultant force f_s = Σ_{j=1}^{m} f_j.
    Compute the acceleration by the force.
    Move this data point for some period.
  endfor
  Compute the error E_DD and iterations I_DD.
endif
```

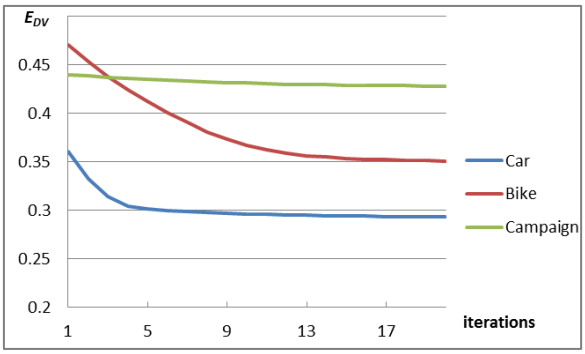

Fig. 5 The change of the data to variable error.

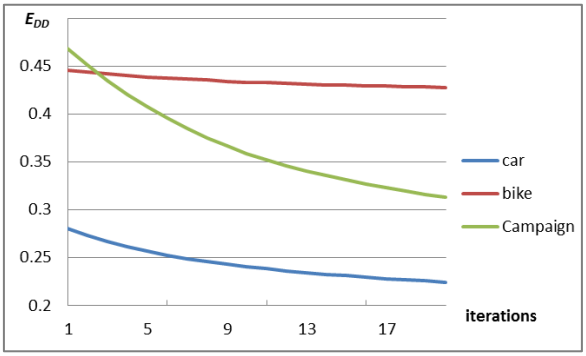

Fig. 6 The change of the data to data error.

scheme is force directed layout [7][5][18]. It iteratively displaces data points until all pair-wise distances in the layout match those in high-dimensional space with minimal error.

In our particular implementation, we construct a network where the vertices correspond to the data points and the edges are springs. (see Fig. 7). Suppose *A, B, C, D, E* are the fixed data points and *P* is the point whose location we plan to adjust. *P* has two types of distances to these five points: (1) the high-dimensional space distances and (2) the 2D layout distances. Their difference forms the error and we should move *P* in a direction that reduces this error the most. The algorithm sets the difference as a force – either drag or push – in each vertex direction. We use f_A, f_B, f_C, f_D and f_E to denote the force vectors from each vertex and they together form an aggregate force as f_S to move *P*. The algorithm is given in Algorithm 3.

The results of this algorithm for our data sets are shown in Fig. 9, fourth column, and the change is listed in Fig. 6. We observe E_{DD} converges. Table 2 reveals this scheme yields large improvements in E_{DD} for the car and the campaign data but not for the bike data.

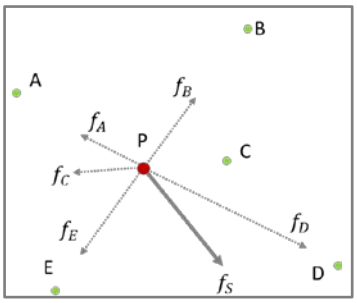

Fig. 7 The force directed adjustment

5.5 Comprehensive Layout (DIFGBC)

The previous sections described the three algorithms {D, I, F} GBC to reduce three types of error. Now, to reduce the overall error, we need to combine them into a single algorithm. The problem is to determine the order of the three algorithms since they can affect each other. In practice we fix the variables first (with DGBC) since this provides a mapping that is more accurate than the one obtained when the mapping error is reduced first. After running DGBC, we have a choice between first moving the data points with respect to the variable (IGBC) and then adjusting the data points with respect to each other (FGBC), or vice versa. There is no clear intuition which order would be better, but for all datasets we tried, the former order gave better results. We therefore use this order – DGBC, IGBC and FGBC – refer as DIFGBC. The final column of Fig. 9 shows the outcome. We observe the layout has inherited improvements from three schemes, but the effects of DGBC are strongest.

Finally, the change of the error is shown in Fig. 8 and Table 2. We can see when DGBC runs, the E_{VV} reduces sharply; then IGBC yields a large improvement of E_{DV}; finally, FGBC reduces the E_{DD}. We can also observe when IGBC and FGBC are running, they will somewhat increase the E_{DD} and E_{DV} respectively. There exists a trade-off – we usually pay more attention to the E_{DV} – run the IGBC first, and then FGBC later for small adjustments. The order of the three adjustments can be altered if user has different priorities.

The E_{VV} has higher error than E_{DV} and E_{DD} since GBC maps the variables to 1D but the other two maps to 2D. But E_{DD} and E_{DV} are also important – they can preserve the accurate data distribution etc.

6 Evaluation

To gauge the quality of a layout, we use the normalized stress metric between L, the matrix of low-dimensional distances L_{ij}, and C, the matrix of high-dimensional distances C_{ij}:

$$stress(L, C) = \sqrt{\frac{\sum_{ij}(L_{ij}-C_{ij})^2}{\sum_{ij} C_{ij}^2}} \quad (2)$$

We use this stress metric to gauge E_{DD}, E_{DV}, and E_{VV}. Each, however, uses a different distance metric for C, set by F in equation (1).

6.1 Data to Data Error

The E_{DD} is due to the difference between L and C. For L and C, we both use Euclidian distance. Now suppose $\|\cdot\|$ is the Euclidean distance. We compute the normalized form of each distance as

$$C_{ij} = F(D_i, D_j) / \sum_{k=1}^{m} F(D_i, D_k)$$
$$L_{ij} = \|P_i - P_j\| / \sum_{k=1}^{m} \|P_i - P_k\|$$

6.2 Data to Variable Error

The GBC plot uses F = (1 - *Value*) for C – a C_{ij} is the j^{th} dimension value of the i^{th} data as $F(D_i, V_j)$. An Euclidian (type) distance for L. However, since the location of the data point is defined by the contour – it uses $\|EP_j - v_i\|$ to represent $F(D_i, V_j)$, we need to use a scale ratio α_{ij} for D_i in the variable V_j

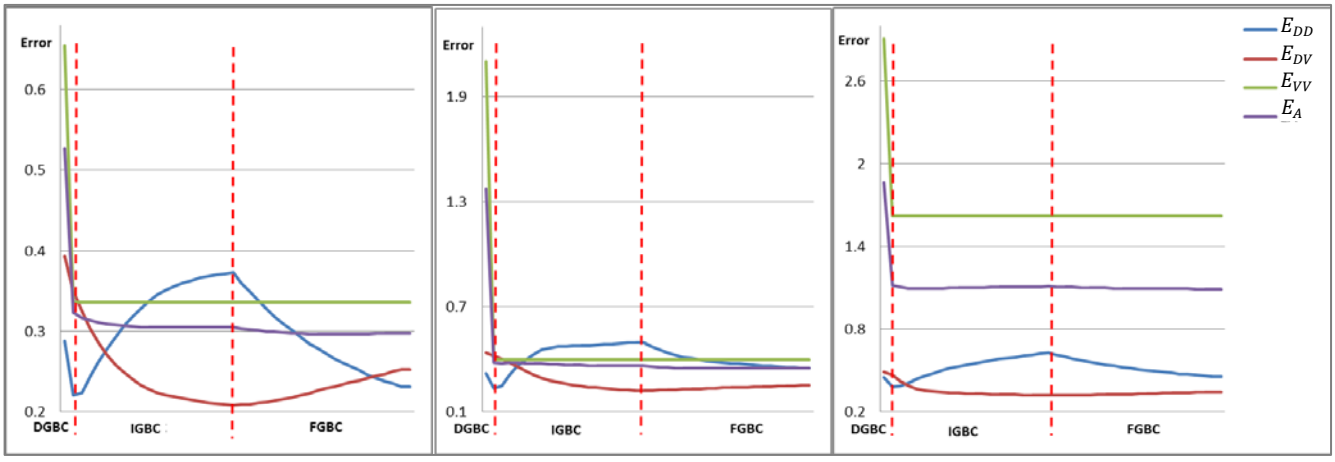

Fig. 8 The error development over the course of the correction.

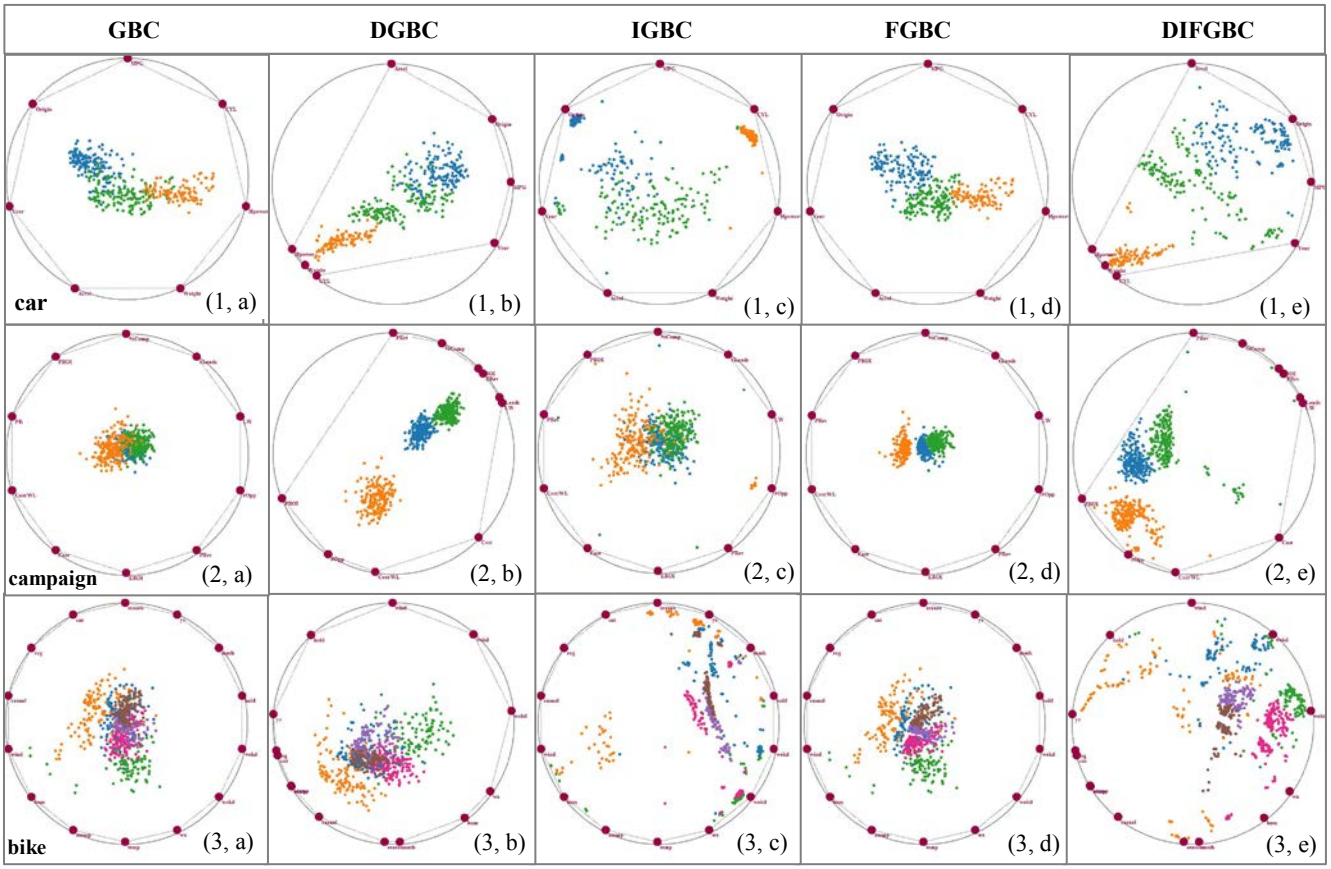

Fig. 9 The GBC error reduction with car data (first line), campaign data (second line) and bike data (third line). The color shows the k-means clusters [9] in the high dimensional distances. The columns are GBC, DGBC, IGBC, FGBC, DIFGBC in order.

$$\alpha_{ij} = \|EP_j - v_i\| / F(D_i,\ V_j) \tag{3}$$

Then the real distance and mapped distance can be obtained as

$$C_{ij} = \alpha_{ij} F(D_i, V_j) \qquad L_{ij} = \|P_i - v_j\|$$

6.3 Variable to Variable Error

This error uses F = 1 - *Correlation* for C. For L, since the GBC places the variables around the circle, we can use the arc length to measure the distance between two variables. The sum of distances of neighboring variables around the circle is its perimeter:

$$\sum_{k=1}^{n} v_k \widehat{v_{N(k)}} = 2\pi r \tag{4}$$

where $v_{N(k)}$ is the neighbor point in the counterclockwise of v_k. However, in the variable to variable distance, we cannot guarantee the sum of the neighbor variables distances satisfies condition (4), so we define a scale ratio β:

$$\beta = 2\pi r / \sum_{k=1}^{n} F\left(V_k, V_{N(k)}\right) \tag{5}$$

Then the real and mapping distance can be obtained as:

$$C_{ij} = \beta F(V_i, V_j) \qquad L_{ij} = \|v_i - v_j\| \text{ (arc length)}$$

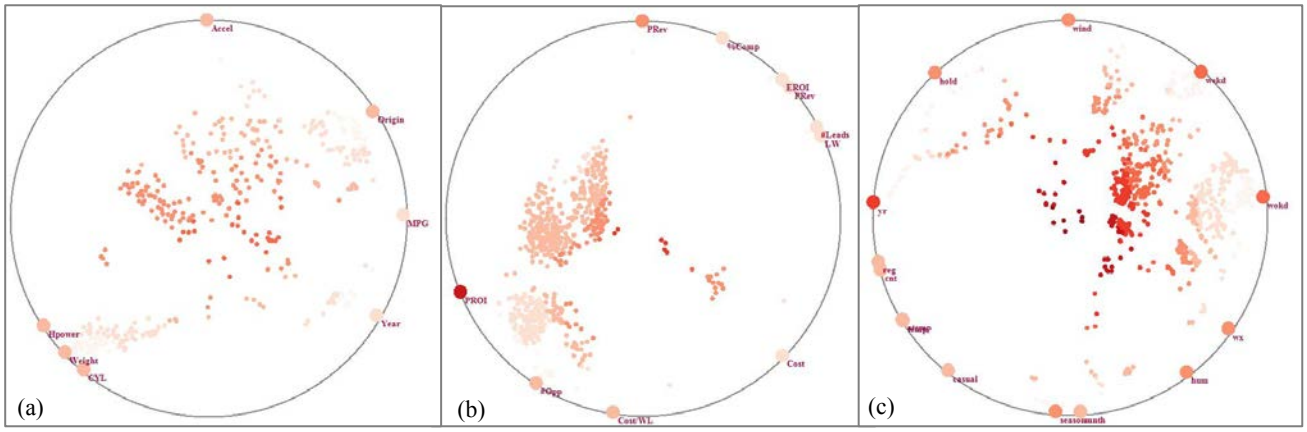

Fig. 10 The error distribution with (a) the car data (b) the campaign data (c) the bike data. Brighter red tones correspond to high value.

6.4 Overall Error

As suggested in Section 5.5, users may have different priorities in the types of distances. Apart from algorithm ordering, we can also express these by giving different weights to the three distances. The overall error is then defined as follows:

$$E_A = w_{DD}E_{DD} + w_{DV}E_{DV} + w_{VV}E_{VV} \quad (6)$$

As was also suggested in Section 5.5, these priorities are likely E_{VV}, then E_{DV} followed by E_{DD} and so we set $w_{DD}: w_{DV}: w_{VV} = 2:4:8$.

6.5 The Error Distribution Display

To give the user an overview of the error distribution, we provide a visualization of it (see Fig. 10). We color each data and variable point as a function of its overall error. In this way the features of the error distribution can be easily identified. From Fig. 10, we can see the data points close to the center of the GBC plot have a high error.

Table 2. The error reduction table

Error	Layout	Car	Campaign	Bike
E_{VV}	GBC	0.65	2.14	2.91
	DGBC	0.34	0.33	1.62
	Reduce	**49%**	**85%**	**44%**
E_{DV}	GBC	0.39	0.44	0.49
	IGBC	0.29	0.43	0.35
	Reduce	**25%**	**2%**	**29%**
E_{DD}	GBC	0.29	0.4	0.45
	FGBC	0.23	0.25	0.43
	Reduce	**28%**	**62%**	**5%**
E_A	GBC	0.53	1.41	1.87
	DIFGBC	0.3	0.31	1.09
	Reduce	**44%**	**78%**	**41%**

7 The GBC Error Explorer

As mentioned before, these types of layouts all suffer from a common problem – the data overlap error (different points have the same position). This kind of error, to the best of our knowledge, is hard to reduce. Springview [1] allows for simultaneously viewing both RadViz and parallel coordinates for view optimization and clutter reduction. We extend this idea and allow users to discover the error by combining different visualization methods into a diagnostic parameter tuning interface – *GBC Error Explorer* (Fig. 1).

Our GBC Error Explorer provides an interactive visual analytics interface to provide the insight. Its interface features the following components: parallel coordinate display, distance heatmap, layout display, error Vis panel and configuration control panel.

The parallel coordinates display provides an overview of the data to help users understand the data values. The distance heat map visualizes the distance matrix. The layout display is the main part of our system – it shows the layout of the distance matrix. The error Vis panel visualizes the error of the layout. Finally, the configuration control panel allows users to manipulate the various parameters.

7.1 The Configuration Control Panel

This control panel contains five major groups: parameter configuration, layout mode, layout color mode, error visualization panel color mode and PC line mode.

The parameter configuration group allows users to set the distance metric for each distance sub-matrix: Euclidean (E), correlation (C), value (V). The layout mode group enables users to choose the layout strategy. The layout color mode error color mode define the color for layout and error respectively. Finally, the PC line mode enables users to pick the line mode of the parallel coordinates.

7.2 The Heat Map Displays

The Distance Heatmap

The heatmap displays on the left visualizes the distance matrix colored using the color bar. We provide the distance matrices *DD*, *DV* and *VV*. However, to lay them as a unit block and maintain the symmetry, we like to add one more sub-matrix *VD* to store the distance of variables to data - same as *DV* (Fig. 1). We find the data are tightly distributed into 3 groups since we find the three blocks.

The Error Vis Panel

The heat map on the right is the error Vis panel. Since the data matrix usually has less variables than data, it is important to know the variables error. Thus, our error Vis panel shows the E_{DV} and E_{VV} vertically with the average ($\overline{E_{DV}}$ and $\overline{E_{VV}}$) on the right border. See Fig. 1. We find the data has higher error with the variable "PROI".

7.3 The Parallel Coordinate Display

The parallel coordinate display shows the data to their dimensions. We have different line modes - straight lines can give users a direct way to see the data, while curve lines are better for the clutter [17].

7.4 Interactions

Our interface provides several types of interactions that manipulate the layout display. Some of these interactions allow users to appreciate the layout errors directly in the display. Others allow filtering operations such as zooming. Considering the different feathers of the data, we apply different techniques to them.

7.4.1 Verification coloring

Distance Color

Fig. 11 a shows our system's capability to visualize the true high-dimensional distances with respect to a user-selected variable (green box) by intensity-shading all points in terms of that distance. An irregular or adverse shading pattern would point to problems, which is not the case for the chosen example. We can clearly see the car has three types of "origin"- from the different color levels.

Error Color

Likewise, Fig. 11 b shows the point-wise layout error with respect to the selected variable (green box). Here we see that the points in the center seem to have a larger layout error– similar conclusion we got from Fig. 10 (a).

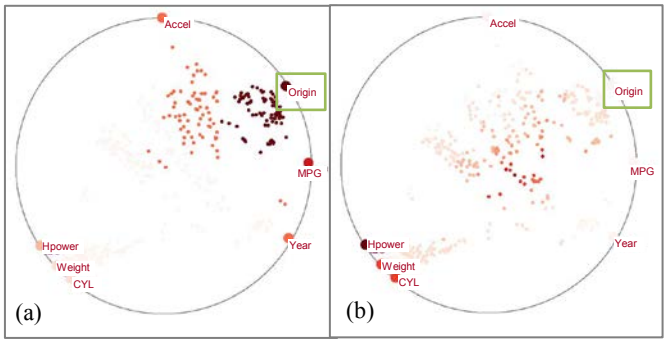

Fig. 11 Verification coloring of (a) distance and (b) error.

7.4.2 Linked displays

Our system also supports linked displays. Users can select a subset of the data in one display and see these in other displays. When we look at the campaign data matrix in the GBC layout, we find they

distribute well as three clear clusters. But we might want to know the details of them. See Fig. 1. We can confirm the yellow group points are close to each other with similar error (from heatmaps) and the variances (from parallel coordinates).

7.4.3 Local layout refinement

The layout can often improve locally if one restricts it to just this region and corresponding high-dimensional subspace. We support two types of local refinements – data-centric and variable-centric.

Data-centric refinement

In the data-centric refinement, the user draws a box in the layout display – such as the green box in Fig. 12a – and then only these data points are included into a focused layout. We saw in Fig. 9c the bike dataset had data points near the center with large error. We select this region and lay out only the points inside it. See Fig. 13b. Now these clusters are much clearer and more defined.

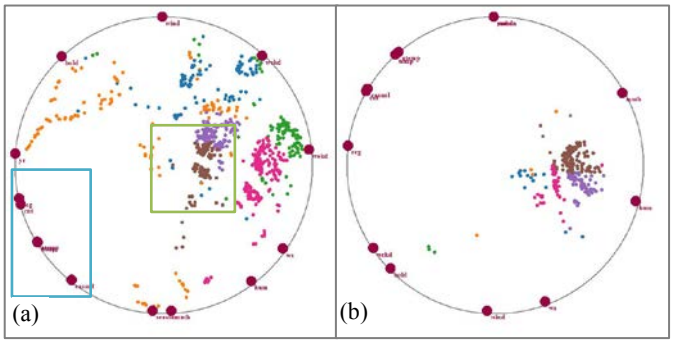

Fig. 12 Data-centric refinement with the bike data matrix. (a) DIFGBC plot and (b) the new layout using just the points inside the green box.

Variable-centric refinement

Conversely, users also draw a box (see the blue one) into the layout display but now only the variables inside this box. This is essentially a subspace selection (see Fig. 12 b). Suppose we wish to know if the bikers are affected by the temperature easily. So we choose a subspace with the related variables, such as temperature, count etc. See Fig. 13. We find the variables form three groups representing the temperature, the number of bike and casual factors respectively.

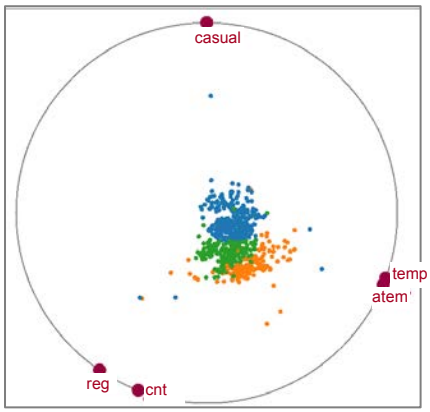

Fig. 13 The variable-centric refinement with the bike data matrix.

8 Conclusion

We have presented a framework that can improve the fidelity of contextual data layouts, in order to better convey the relations of data items and data attributes. We first unified the different data layouts in this class of visualization algorithms, choosing the GBC plot as the standard formulation. We then proposed three algorithms – distance spaced layout, iterative error reduction and force directed adjustment – to reduce the error. We also developed an interface by which users can explore the error with interactions.

In this current work we have focused on contextual layouts in which the attributes (variables) are arranged at the periphery of the data points. While separating the variables and data points makes for a structured display, better optimizations might be achievable by allowing the attribute points to mingle with the data points. This is subject of current research efforts.

Acknowledgments

This research was partially supported by NSF grant IIS 1117132 and the MSIP, Korea, under the "IT Consilience Creative Program (ITCCP)" (NIPA-2013-H0203-13-1001) supervised by NIPA. We thank the reviewers, in particular the primary, for their diligent comments that greatly improved the presentation of the work.

References

[1] E. Bertini, L. Aquila, G. Santucci. "SpringView: Cooperation of Radviz and Parallel Coordinates for View Optimization and Clutter Reduction." *Proc. of IEEE International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV)*, 2005.

[2] J. Chambers, W. Cleveland, P. Tukey, *Graphical Methods for Data Analysis,* Duxbury Press, 1983.

[3] G. Grinstein, M. Trutschl, U, Cvek, "High-dimensional visualizations," *Proc. Visual Data Mining Workshop, KDD*, 2001.

[4] J. Hartigan, "Printer Graphics for Clustering," *J. Statistical Computation and Simulation,* 4(3): 187-213, 1975.

[5] S. Ingram, T. Munzner, M. Olano, "Glimmer: Multilevel MDS on the GPU". *IEEE Trans. Vis. Comput. Graph*. 15(2): 249-261 (2009)

[6] A. Inselberg, B. Dimsdale, "Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry," *Proc. IEEE Visualization*, pp. 361-378, 1990.

[7] A.Morrison, G. Ross, M. Chalmers, "Fast multidimensional scaling through sampling, springs and interpolation." *Information Visualization* 2(1):68-77 (2003)

[8] L.Maaten, G.Hinton, "Visualizing data using t-SNE", *J. Machine Learning Research*, 9: 2579-2605, 2008

[9] MacQueen, J. B. (1967). "Some Methods for classification and Analysis of Multivariate Observations". *Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press. pp. 281–297. MR 0214227. Zbl 0214.46201.

[10] E. Kandogan, "Star Coordinates: A Multi-Dimensional Visualization Technique with Uniform Treatment of Dimensions," *Proc. IEEE Information Visualization, Late Breaking Topics*, pp. 9-12, 2000

[11] J. Kruskal. M. Wish, *Multidimensional Scaling*. Sage Publications, 1977.

[12] J. Nam, K. Mueller, "TripAdvisorN-D: A Tourism-Inspired High-Dimensional Space Exploration Framework with Overview and Detail," *IEEE Trans. Visualization and Computer Graphics*, 19(2):291-305, 2013.

[13] M. Meyer, A. Barr, H. Lee, M. Desbrun, "Generalized Barycentric Coordinates on Irregular Polygons," *J. Graphics Tools*, 7(1), 2002.

[14] P. Hoffman, G. Grinstein, K. Marx, I. Grosse, E. Stanley, "DNA Visual and Analytic Data Mining", *Proc. IEEE Vis*, pp. 437-441, 1997.

[15] K. Hinum, S. Miksch, W. Aigner, S. Ohmann, C. Popow, M. Pohl, M. Rester, "Gravi++: Interactive Information Visualization to Explore Highly Structured Temporal Data," *J. Universal Computer Science* 11(11):1792-1805, 2005.

[16] J. Yi, R. Melton, J. Stasko, J. Jacko, "Dust & Magnet: Multivariate Information Visualization using a Magnet Metaphor," *Information Visualization,* 4(4) : 239-256, 2005.

[17] H.Zhou, X.Yuan, H.Qu, W.Cui, B.Chen: Visual Clustering in Parallel Coordinates. *Comput. Graph. Forum* 27(3): 1047-1054 (2008)

[18] Z. Zhang, K. McDonnell, K. Mueller, "A Network-Based Interface for the Exploration of High-Dimensional Data Spaces," *Proc. IEEE Pacific Vis*, Songdo, Korea, pp. 17-24, 2012.