Info Vis

Article

Information Visualization
2019, Vol. 18(1) 3–32

DOI: 10.1177/1473871617715860
journals.sagepub.com/home/ivi
SAGE

Adjustable general line coordinates for visual knowledge discovery in n-D data

Boris Kovalerchuk[1] and Vladimir Grishin[2]

Abstract

Preserving all multidimensional data in two-dimensional visualization is a long-standing problem in Visual Analytics, Machine Learning/Data Mining, and Multiobjective Pareto Optimization. While Parallel and Radial (Star) coordinates preserve all n-D data in two dimensions, they are not sufficient to address visualization challenges of all possible datasets such as occlusion. More such methods are needed. Recently, the concepts of *lossless General Line Coordinates* that generalize Parallel, Radial, Cartesian, and other coordinates were proposed with initial exploration and application of several subclasses of General Line Coordinates such as Collocated Paired Coordinates and Star Collocated Paired Coordinates. This article explores and enhances benefits of General Line Coordinates. It shows the ways to increase expressiveness of General Line Coordinates including decreasing occlusion and simplifying visual pattern while preserving all n-D data in two dimensions by adjusting General Line Coordinates for given n-D datasets. The adjustments include relocating, rescaling, and other transformations of General Line Coordinates. One of the major sources of benefits of General Line Coordinates relative to Parallel Coordinates is twice less number of point and lines in visual representation of each n-D points. This article demonstrates the benefits of different General Line Coordinates for real data visual analysis such as health monitoring and benchmark Iris data classification compared with results from Parallel Coordinates, Radvis, and Support Vector Machine. The experimental part of the article presents the results of the experiment with about 70 participants on efficiency of visual pattern discovery using Star Collocated Paired Coordinates, Parallel, and Radial Coordinates. It shows advantages of visual discovery of n-D patterns using General Line Coordinates subclass Star Collocated Paired Coordinates with n = 160 dimensions.

Keywords

Multidimensional data visualization, knowledge discovery, visual data mining, machine learning, general line coordinates, lossless visual representation, reversible visual representation, adjustable coordinates, clutter reduction, parallel coordinates

Introduction

Discovering patterns in multidimensional data using visual means is a long-standing problem in Data Science.[1–7,50–51] This article focuses on lossless visual representation on n-D data in 2D. While the term "lossless visualization" (attributed to Parallel Coordinates (PC)) can be traced at least from 1997[8] to 2015,[9] unfortunately, to the best of our knowledge, only a few new lossless visual representations have been developed for the last 25 years; this includes PC summarized in Inselberg[3] and special Star Coordinates[10,11] to solve this difficult and long-standing problem.

[1]Central Washington University, Ellensburg, WA, USA
[2]View Trends, Ltd, Port St. Lucie, FL, USA

Corresponding author:
Boris Kovalerchuk, Central Washington University, 400 East University Way, Ellensburg, WA 98926-7501, USA.
Email: borisk@cwu.edu

The concept of *lossless General Line Coordinates* (GLC) that generalizes Parallel, Radial (Star), and other coordinates was proposed with initial exploration and application of several subclasses of GLC such as Collocated Paired Coordinates (CPC) and Star CPC in Kovalerchuk,[12] Grishin and Kovalerchuk,[13] Kovalerchuk and Grishin,[14] Kovalerchuk and Smigaj,[15] and Kovalerchuk.[16] The GLC with their subclasses significantly expand a set of lossless coordinate systems. GLC contain an *infinite number of coordinate systems*. Those studies demonstrated the efficiency of some GLC on World hunger data, Challenger disaster, Natural Language Processing (humor detection), and modeled data. GLC allow preserving (1) reversible (one-to-one) mapping between n-D points and their 2D representations and (2) similarity of n-D points in their 2D visual representations (similar n-D points are represented as similar graphs in the visualization space). The main motivation for lossless GLC is preserving all n-D data in 2D visualization, that is, the abilities to restore each n-D point completely from its 2D visual representation that is useful and important in Visual Analytics, Machine Learning/Data Mining, and Multiobjective Pareto Optimization.

While Parallel and Radial coordinates preserve n-D data, they are not sufficient to address visualization challenges of all possible datasets such as occlusion that is common for larger datasets. More lossless methods are needed. In contrast with PC, a GLC subclass of Parametric Shifted Paired Coordinates (PSPC) allows representing the n-D point of interest losslessly (e.g. the center of the class or cluster C) as a single 2D point C*[16] as explained in Appendix 1.

This single 2D point property leads to another math property that all PSPC 2D graph representations of n-D points of a given class will be in the vicinity of 2D point C* if respective n-D points are in the vicinity of n-D point C. Parallel and Radial Coordinates do not have this capability. In PC, no n-D point can be represented as a single 2D point with such preserved n-D vicinity. This property significantly simplifies the visual pattern of the n-D data class and deeply improves the abilities to discover n-D classes visually using their 2D PSPC representations along with two times less occlusion than for graphs in PC.[16] The last property is a result of two times less nodes in PSPC graphs. Next, this PSPC property decreases the significance of attribute order for class pattern discovery that is important in PC.[17,18] Lossless methods can be naturally combined with other dimension-reduction methods such as principal component analysis (PCA). The PCA can result in a very lossy visual representation when only two principal components are used to visualize n-D data out of n principal components. In contrast, PCA often provides a very good approximation of n-D data when more principal components are preserved, say 30 out of 100 instead of 2 of them for a given n-D data. The 30D points in these principal components can be visualized losslessly by GLC and the original 100D points, respectively, will be visualized with acceptable loss of information.

Many other 2D representations[19–26] belong to the class of lossy irreversible visual 2D representations with shortages illustrated for several of them in multiple examples.[27–36] The difference between lossy and lossless methods is important from three critical aspects: (1) scientific rigor, (2) efficiency of applications, and (3) social and ethical norms, because the loss and corruption of data can lead to degradation in each of them. However, while lossy methods suffer from data *corruption* and *difficulties to interpret* results, lossless methods suffer more from *occlusion*.[36,37] While these methods are powerful, they do not provide a "silver bullet" for all possible tasks and datasets and have relatively *complex* 2D graph representation. Therefore, the major goal of this article is *decreasing the impact of occlusion* and *simplifying graphs*. We approach this challenge by developing adaptable GLC methods in contrast with the typical approach where Parallel and Radial Coordinate methods are not modified for given data.

This article expands the prior GLC research in the following aspects: (1) proposing new methods for decreasing occlusion; (2) simplifying visual patterns for classification tasks; (3) demonstrating on real Iris and health monitoring data the efficiency of new compact lossless representation by PSPC; (4) proposing a new two-layer GLC concept and demonstration of its efficiency on real data; (5) demonstrating advantages of closed contour lossless visual representations over PC for high-dimensional data in the experiment with about 70 participants for classification of modeled data (linear hypertubes in subspaces); (6) clarifying limits of high-dimensionality of data for human visual classification of modeled n-D data (linear hypertubes in subspaces) in PC, Star CPC, and Radial Coordinates; (7) establishing visual representation of linear relations in various GLC and showing their advantages over PC; and (8) comparing and combining GLC with stick figures (SF) to cover larger dimensions.

The rest of this article is organized as follows. Section "Background definitions" presents the background information from prior publications that include definitions of general line coordinates with examples of GLC subclasses. Section "Methods for decreasing occlusion and pattern simplification" presents the new methods of GLC adjustment for decreasing occlusion and visual pattern simplification with the preservation of losslessness of 2D representation. Section "Advantages of GLC for real data visualization"

shows the advantages of different GLC in real data representations. It includes health monitoring and Iris data classification compared with results from PC, Radvis, and Support Vector Machine (SVM). Section "Experimental comparison of visual recognition of 160D linear data structures with Gaussian noise in PC, Stars, and CPC Stars" presents the experimental results of visual pattern discovery on 160D data using GLC by about 70 participants involved. Section "Visual representation of n-D relations in GLC" presents the visual representation of n-D relations in GLC, and section "Comparison of GLC with SFs" compares and combines the GLC with SFs. Section "Conclusion" summarizes the results and outlines the future research, and Appendix 1 presents the results from background publications for the convenience of self-contained reading of the article.

Background definitions

Below we present the main GLC concepts followed[16] by more details in Appendix 1. Table 1 summarizes the different forms of GLC. The GLC class contains the well-known Parallel and Radial (Star) coordinates, and the new ones are listed in Table 1, which generalize them by locating coordinates in any place, direction, and in any topology (connected or disjoined).

The examples of GLC are shown in Figures 1–3, starting from PC in Figure 1(a). *In-Line Coordinates* (ILC) shown in Figure 2(d) are similar to PC, except that the axes $X_1, X_2, ..., X_n$ are horizontal, not vertical. Each pair (x_i, x_{i+1}) is represented as a Bezier curve. The height of the curve is the distance between the two adjacent values, $d(x_i, x_{i+1})$.

The algorithm for representing n-D points in 2D using lossless *CPC* is presented below. Consider as an example a 6D state vector $\mathbf{x} = (x, y, x', y', x'', y'')$, where x and y are the location of the object, x' and y' are the velocities, and x'' and y'' are the accelerations of an object. The main steps of the algorithm are as follows:

- Normalization of all dimensions to some interval, for example, [0, 1];
- Pairing attributes into consecutive pairs (x, y), (x', y'), (x'', y'');
- Plotting each pair in the same orthogonal normalized Cartesian coordinates X and Y;
- Plotting a directed graph $(x, y) \rightarrow (x', y') \rightarrow (x'', y'')$.

Figure 2(a) shows the application of this algorithm to a 6D vector (5, 4, 0, 6, 4, 10) with the directed graph drawn as two arrows: from (5, 4) to (0, 6) and from (0, 6) to (4, 10).

The *Shifted Paired Coordinates* (SPC) show each next pair in the shifted coordinate system. The first pair (5, 4) is drawn in the (X, Y) system. The next pair (0, 6) is drawn not in the original system (X, Y), but in the shifted coordinate system denoted as (X + 1, Y + 1), where coordinate X is shifted up by 1 and coordinate Y is shifted to the right by 1. This means

Table 1. Line coordinates.

Type	Characteristics
General Line Coordinates (GLC)	Drawing n-coordinate axes in 2D in a variety of ways: curved, parallel, unparalleled, collocated, disconnected, etc.
Collocated Paired Coordinates (CPC) in 2D	For each n-D point **x** splitting it into *pairs* of its coordinates (x_1, x_2), ..., (x_{n-1}, x_n); drawing each pair as 2D point in the same two axes on the plane and linking these 2D points to form a directed graph.
Collocated Paired Coordinates in 3D	For each n-D point **x**, splitting n-coordinates into triples and representing each triple as 3D point in the same three axes, and linking these points to form a directed graph for each n-D point.
Shifted Paired Coordinates (SPC)	Drawing each next pair in the shifted coordinate system by adding (1, 1) to the second pair, (2, 2) to the third pair, $(i-1, i-1)$ to the i-th pair, and so on. More generally, shift can be a function of some parameters.
Anchored Paired Coordinates (APC)	Drawing each next pair in the shifted coordinates relative to coordinates shifted to the location of the first pair (x_1, x_2) of a given n-D point **x**.
Partially Collocated Coordinates	Drawing some coordinate axes in 2D collocated and some coordinates not collocated.
Partially Collocated Radial Coordinates	Drawing some radial coordinate axes in 2D collocated and some coordinates not collocated.
In-Line Coordinates (ILC)	Drawing all coordinate axes in 2D located one after another on s-single straight line.
Circular and n-gon coordinates	Drawing all coordinate axes in 2D located on a circle or a n-gon one after another.

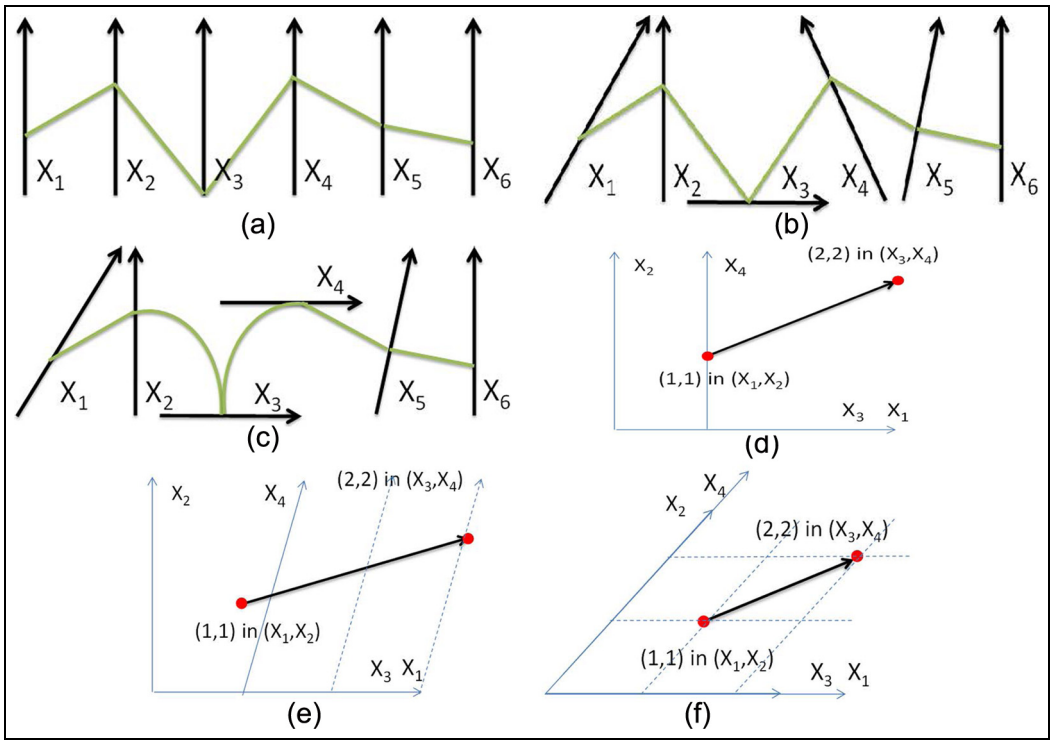

Figure 1. Examples of General Line Coordinates. (d–f) 4D point (1, 1, 2, 2) in different coordinate systems; (a) 6D point in Parallel Coordinates, (b) 6D point in General Line Coordinates with straight lines, (c) 6D point in General Line Coordinates with curves, (d) Partially Collocated Orthogonal (Ortho) Coordinates, (e) Partially Collocated Ortho non-Ortho Coordinates, and (f) Collocated non-Ortho Coordinates.

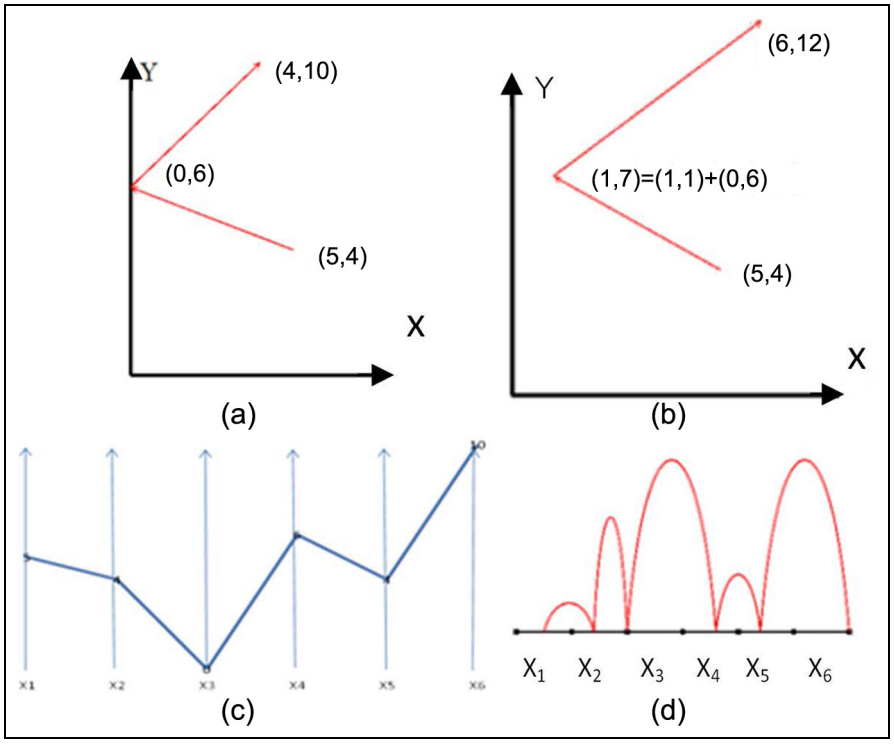

Figure 2. 6D data point (5, 4, 0, 6, 4, 10) in different coordinate systems: (a) Collocated Paired Coordinates, (b) Shifted Paired Coordinates, (c) Parallel Coordinates, and (d) In-Line Coordinates (ILC).

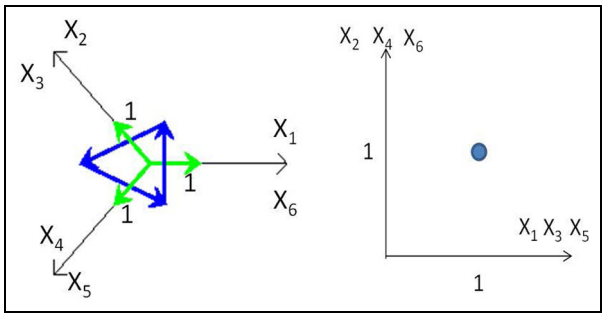

Figure 3. 6D point (1, 1, 1, 1, 1, 1) in two X_1–X_6 coordinate systems (left —in Radial Collocated Coordinates, right—in Cartesian Collocated Coordinates).

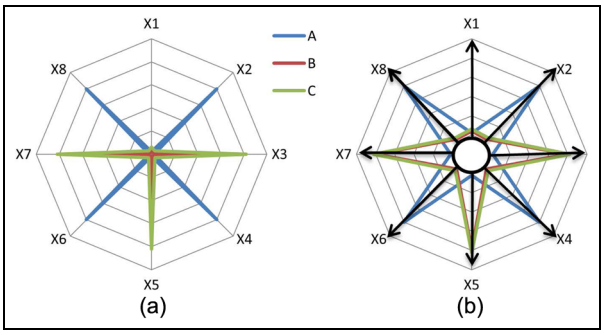

Figure 4. Occlusion removal demonstration: 8D points A, B, and C in (a) traditional radial coordinates and (b) Unconnected Radial Coordinates; (a) values of 8D point A occluded by 8D points B and C near the origin due to connectedness of coordinates at the origin and (b) values of 8D point A are not occluded by 8D points B and C due to unconnectedness of coordinates.

that the pair (0, 6) in coordinates (X + 1, Y + 1) will be a pair (0, 6) + (1.1) = (1, 7) in the original coordinates (X, Y). For shift n and coordinates (X + n, Y + n), it is $(a, b)_{(X+n,\ Y+n)} = (a + n, b + n)_{(X,\ Y)}$. The pair (4, 6) is drawn in the (X + 2, Y + 2) coordinates. For point (5, 4, 0, 6, 4, 10), the graph includes the arrows: from (5, 4) to (1, 1) + (0, 6) = (1, 7) then from (1, 7) to (2, 2) + (4, 10) = (6, 12) (see Figure 2(b)).

The *Anchored Paired Coordinates* (APC) represent next pairs shifted relative to the first pair that serves as an "anchor." In the example above, the pairs (x′, y′) and (x″, y″) are represented as vectors that start at the anchor point (x, y) with plotting vectors ((x, y), (x + x′, x + y′)) and ((x + x′, x + y′), (x + x″, x + y″)) as arrows.

Figure 3 shows the graph of 6D point (1, 1, 1, 1, 1, 1) in *Partially Collocated Radial Coordinates* on the left as a blue triangle. The same 6D point in the *Cartesian CPC* on the right gives a much simpler graph as a single point. Figure 3 illustrates the perceptual and cognitive differences between alternative 2D representations of the same n-D data, where a 2D point is much simpler perceptually and cognitively than a triangle.

Methods for decreasing occlusion and pattern simplification

This section describes the methods for decreasing occlusion and pattern simplification in different GLC by shifting, relocating, and scaling coordinates. These transformations are applied to Radial, Parallel, Shifted Paired, Circular, and n-gon coordinates.

Decreasing occlusion by shifting coordinates

In *Radial Coordinates*, the different n-D data points *occlude* each other, when their values are close to the common coordinate origin, because that area is small. Figure 4(a) illustrates this occlusion where it is impossible to see the full difference between three 8D points shown as red, green, and blue lines. The *Unconnected Radial Coordinates* (URC) shown in Figure 4(b) resolve this occlusion issue by starting all coordinates at the edge of the circle instead of the common origin, that is, by *shifting* all coordinates to that edge. Thus, more freedom in locating coordinates shows its benefits to decrease the occlusion in Radial Coordinates.

The same origin-based occlusion takes place in the Cartesian Coordinates, Collocated Cartesian, and Collocated Star Coordinates because all of them have a common origin of all coordinates. The *way to resolve* this origin-based occlusion is the same as for Radial Coordinates—shifting coordinates from the common origin along their directions, that is, making these coordinates *unconnected*. PC are free from this occlusion problem due to the absence of the common origin.

For *PC* shifts of coordinates allow revealing visual patterns *faster* and make patterns *simpler* by presenting them as preattentive straight lines as we show below. It exploits a well-known property that straight lines are *preattentive features*.[38,39] A summary of Appelbaum and Norcia[39] is presented in Appendix 1. Consider two 6D data points A = (0.3, 0.6, 0.4, 0.8, 0.2, 0.9) in blue and B = (0.35, 0.68, 0.48, 0.85, 0.28, 0.98) in orange in Figure 5. Figure 5(a) shows A and B in a standard PC as non-preattentive zig-zag lines. In contrast to Figure 5(b) and (c), A is a preattentive straight line and B is much simpler than in Figure 5(a). The lines in Figure 5(b) and (c) can be easily compared and correlated. This simplification was achieved by changing PC to Shifted Coordinates.

The paired lossless representation of 6D point A in Figure 6(a) is not only a preattentive horizontal straight line but also twice simpler than in PC having only three 2D points versus six 2D points in PC in

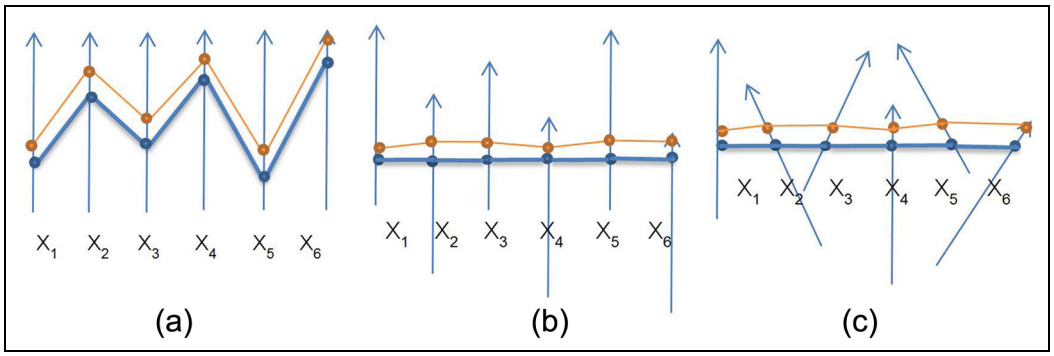

Figure 5. Non-preattentive versus preattentive visual representations (linearized patterns): 6D point A = (3, 6, 4, 8, 2, 9) in blue and 6D point B = (3.5, 6.8, 4.8, 8.5, 2.8, 9.8) in orange in Traditional and Shifted Coordinates; (a) data in traditional Parallel Coordinates—non-preattentive representation, (b) data in shifted Parallel Coordinates—preattentive representation, and (c) data in shifted General Line Coordinates.

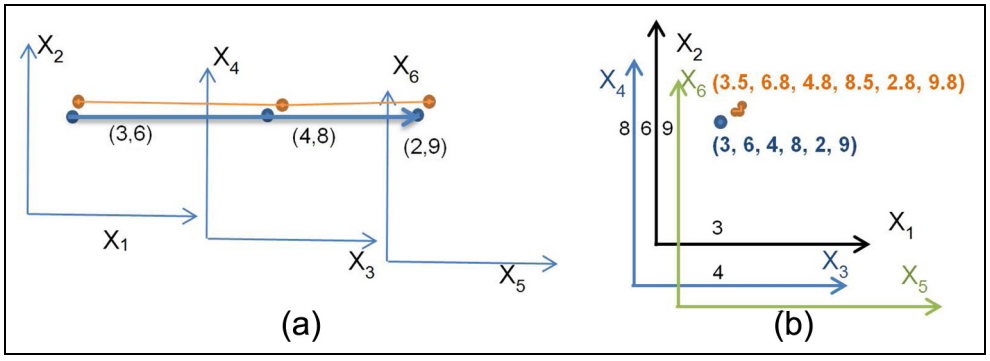

Figure 6. Preattentive lossless representation of 6D point A = (3, 6, 4, 8, 2, 9) in blue and simplified representation of point B = (3.5, 6.8, 4.8, 8.5, 2.8, 9.8) in orange in Shifted Paired Coordinates. (a) 6D point A as a preattentive horizontal straight line and (b) 6D point A as a preattentive single 2D point.

Figure 5(a). Figure 6(b) represents the same 6D point A as a single 2D point losslessly and preattentively.

Simplifying patterns by relocating and scaling coordinates

Shifted PC is a visual way to implement the idea of designing a *complex non-linear transform* of the n-D data space into another space where a linear discriminant function or a hyperplane can be built for n-D data classification. This linearization idea is behind the algorithm based on *rescaling*.[53] Shifting coordinates in PC to get a linear representation is similar to applying the *Rescaled PC*.[40] Rescaling visually can be done without shifting, but by contracting or expanding coordinates as it is shown in Figure 7. Note that rescaling may change perception because the resolution of some coordinates can decrease.

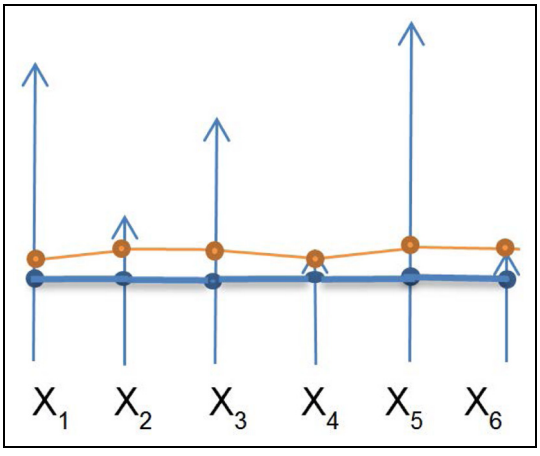

Figure 7. Example of preattentive linearized patterns of 6D data in scaled Parallel Coordinates.

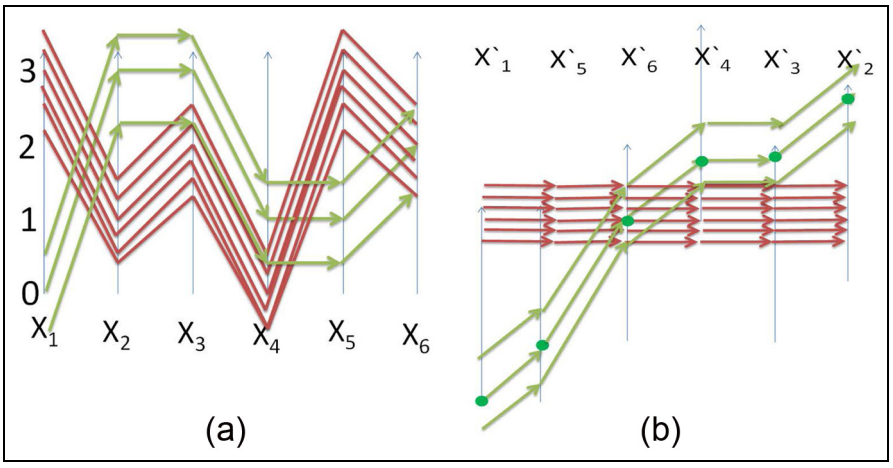

Figure 8. Simplification of visual representation by shifting and reordering Parallel Coordinates: (a) original visual representation of two classes in Parallel Coordinates and (b) simplified visual representation after shifting and reordering Parallel Coordinates.

Rescaling in PC can be done with minimal visual changes in the coordinates by changing the number of pixels used for the units in a coordinate. Shrinking the coordinate line of the given length leads to shrinking the units of that coordinate (decreasing the number of pixels devoted to the scale unit).

The same decreased number of pixels can be implemented by keeping the length of the coordinate. In this case, the range of the values that the coordinate line carries will be larger. This would require redrawing the dividers of units on the coordinate making them denser.

Figure 8 illustrates the simplification of visual representation of n-D data of two classes by *shifting and reordering* PC. In Figure 8(b), some coordinates are moved up and some others moved down. The order of coordinates is also changed. As a result, the first class (red) became preattentive being represented by horizontal straight lines. In Figure 8(b), the second class (green) became simpler too. It is now a set of monotone increasing lines that are easier and faster recognizable than zig-zag lines in Figure 8(a).

Figure 9 shows the ways to simplify visualization of other GLC using shifting and rescaling coordinates. In Figure 9, this approach is applied to the traditional Radial Coordinates, and new circular and n-gon coordinates that were introduced in Kovalerchuk[12] without these simplifications. For comparison, the same data are shown in Figure 9(a) in PC and in Figure 9(c) in Radial Coordinates.

In the Circular Coordinates (Figure 9(b)), the circle is divided into segments and each segment encodes a coordinate (e.g. in a normalized scale within [0, 1]), where each x_i of an n-D point $\mathbf{x}$ is located in the respective coordinate X_i. For instance, $x_1 = 0.3$ in Figure 9(b) is located at the respective distance from the origin of X_1 along the X_i segment of the circle.

Next, these points x_i are connected to form the directed graph starting from x_1. Connecting x_1 and x_n leads to a closed contour (see dotted lines in Figure 9(d)). It is advantageous perceptually, which we will illustrate later in detail. Closed contours are colored to distinguish n-D points and their classes. Similarly, the n-gon (triangle, rectangle, pentagon, and so on) is divided into segments and each segment encodes a coordinate.

These points are connected to form a graph (see Figure 9(g)). Figure 9(a)–(c) and (g) shows that Circular Coordinates and n-gon coordinates have the same complexity (require the same number of lines) as known Parallel and Radial Coordinates for the same 4D point. Therefore, these new coordinates have a potential to be successful in the same types of applications where Parallel and Radial Coordinates have been successful.

The idea of simplification is transforming the original 2D graph from irregular shape to a *familiar regular symmetric shape* such as a square, pentagon, and hexagon. Figure 9(d)–(f) shows the results of such transformation of an irregular rectangle to a square for 4D point A in Circular Coordinates, and Figure 9(g)–(i) shows this in n-gon (square) coordinates and Radial Coordinates. All these transformations are linear. Under these transformations respective segments of the circle shrink to subsegments shown in Figure 9(d) and (e). In Figure 9(d), this resulted in moving $x_1 = 0.3$ down and $x_4 = 0.2$ up to form a horizontal line. At the position of 0.5, moving both x_1 and x_4 to the location of 0.5 on respective coordinates ensures getting a square because x_2 and x_3 are already in those 0.5 positions.

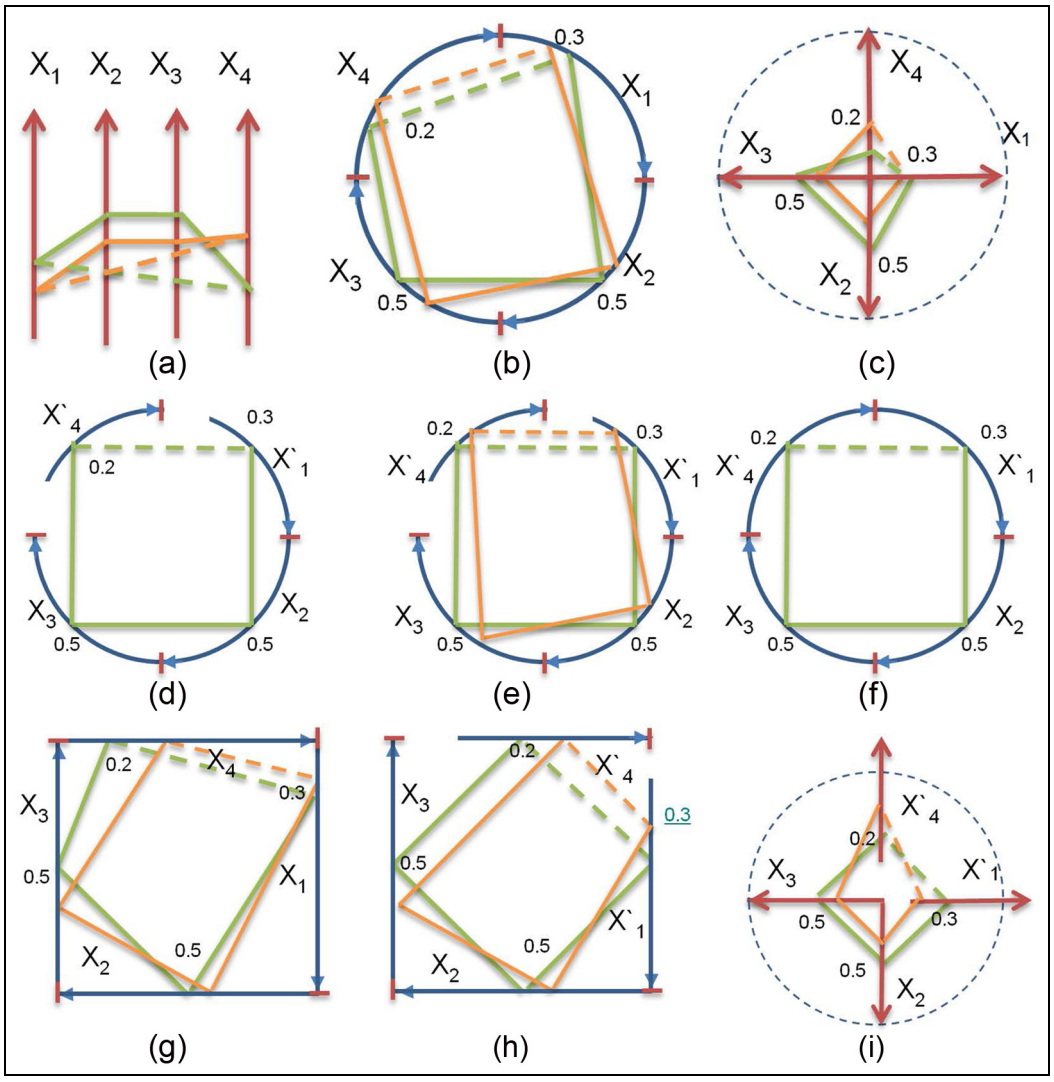

Figure 9. Visualization simplification by relocating and rescaling coordinates for 4D points A = (0.3, 0.5, 0.5, 0.2) in green and B = (0.2, 0.45, 0.4, 0.4) in orange displayed in Parallel, Circular, Radial, and n-gon (square) coordinates fully or partially connected; (a) two 4D points in Parallel Coordinates, (b) two 4D points in Circular Coordinates, (c) two 4D points in Radial Coordinates, (d) 4D point A in Contracted Circular Coordinates, (e) two 4D points in Contracted Circular Coordinates, (f) 4D point A in Disproportional Circular Coordinates, (g) two 4D points in n-gon (square) coordinates, (h) two 4D points in Partially Connected n-Gon Coordinates, and (i) two 4D points in Partially Connected Radial Coordinates.

These linear transformations map X_1 to X_1' with $0 \rightarrow b$, $0.3 \rightarrow 0.5$, $1.0 \rightarrow 1.0$, and X_4 to X'_4 with $0 \rightarrow d$, $0.2 \rightarrow 0.5$, and $1.0 \rightarrow 1.0$. The linear equations to find these transforms are $y = ax + b$ and $y = cx + d$, where for $y = ax + b$ and X_1 we have two pairs ($x = 0.3$, $y = 0.5$) and ($x = 1$, $y = 1$). This leads to a system of two linear equations: $0.5 = 0.3a + b$ and $1 = a + b$, with a solution: $a = 0.714$ and $b = 0.286$. Thus, coordinate X_1' starts at the location 0.286 on the circle segment X_1. In polar coordinate, this location is given by a pair (R, α). Similar computations are conducted for $x_4 = 0.2$ on coordinate X_4. In contrast to Figure 9(f), the shape is simplified by a non-linear monotone transform that maps X_1 to X_1' with $0 \rightarrow 0$, $0.3 \rightarrow 0.5$, $1.0 \rightarrow 1.0$, and maps X_4 to X'_4 with $0 \rightarrow 0$, $0.2 \rightarrow 0.5$, and $1.0 \rightarrow 1.0$. It can be modeled by a piecewise linear transformation with two linear parts that can be found by solving two systems of two linear equations.

The scaling approach without changing the length and location of PC was implemented for PC in Theus.[40] Above this, idea was generalized for various GLC allowing relocation and resizing of coordinates in addition to rescaling. The advantaged of rescaling was shown in Theus[40] for n-D data for Tour de France 2005 with a conclusion that after each axis is

aligned at the individual medians, the display clearly reveals the most information.

In terms of the examples in Figure 9, the role of the linearized n-D point A is played by a set of individual medians in each axis in Theus.[40] Theus[40] proposed several options for selecting such an n-D point: the mean, the median, a specific case, or a specific value. We will call this n-D point an *alignment n-D point*. He also noted that alignment of coordinates can be controlled without using a particular n-D point by either individually scaling the axes or by using some common scale over all axes with a general conclusion: "Parallel coordinate plots are not very useful 'out of the box', i.e., without features like α-blending and scaling options."

The same simplification approach as described above can be applied for Circular and n-gon coordinates for a higher number of dimensions with substituting squares to pentagons, hexagons, and other respective n-gons. It is likely that the highest dimensions will be comparable with the dimension that is workable for Radial Coordinates. It is illustrated in Figure 9 where Figure 9(c) and (i) shows the same two n-D points in Radial Coordinates and Partially Connected Radial Coordinates.

Circular, n-gon coordinates, and Radial Coordinates use the same number of 2D points around a common center to represent each n-D point. The difference is in the way how the points are located. We also expect that easiness for people to identify the clusters when they look at data points in Circular and n-gon coordinates should be similar to it for Radial Coordinates for the same reason. Section "Experimental comparison of visual recognition of 160D linear data structures with Gaussian noise in PC, Stars, and CPC Stars" and Grishin and Kovalerchuk[13,14] show that the dimensions up to n = 192 are possible for the Radial Coordinates. The further increase in the dimension is the topic of the future studies.

Advantages of GLC for real data visualization

This section demonstrates advantages of the methods for decreasing occlusion and pattern simplification in GLC in real-world tasks of health monitoring and Iris data classification. It involves SPC, two-layer visual representation, and comparison with alternative methods such as PC, Radvis, and SVM.

Health monitoring

Figure 10 illustrates the opportunities of using PSPC for health monitoring of an individual in comparison with PC. In this example, four health characteristics are monitored: systolic blood pressure, x_1; diastolic blood pressure, x_2; pulse, x_3; and total cholesterol, x_4. At the initial moment, the individual health status is presented by four values of these characteristics (100, 150, 95, 250). A desired health status for these characteristics is identified as (70, 120, 60, 190). The goal is to monitor the progress the individual is making

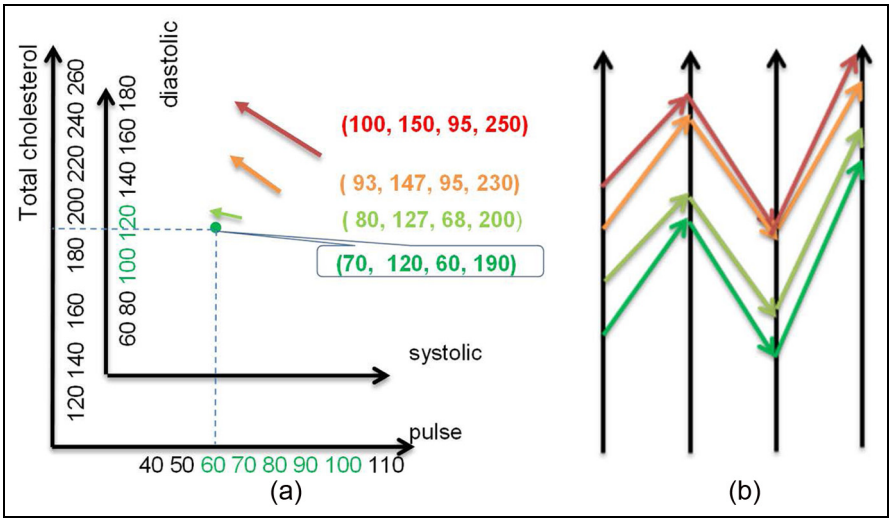

Figure 10. 4D health monitoring visualization in (a) PSPC and (b) Parallel Coordinates with parameters: systolic blood pressure, diastolic blood pressure, pulse, and total cholesterol at four time moments; (a) PSPC: the green dot is the desired goal state, the red arrow is the initial state, the orange arrow is the health state at the next monitoring time, and the light green arrow is the current health state of the person and (b) Parallel Coordinates.

toward this goal with a complex of medical treatments, diet, exercises, and so on.

In Figure 10(a), in PSPC, the goal is presented as a single preattentive point that is simple metaphor to learn because targets quite often are represented as points. This single point is a result of the PSPC design. In contrast, the PC show the goal as a zig-zag line with four points that is not preattentive. Next, the current status in Figure 10(a) for PSPC consists just of a single line (arrow). In PC, it is again a zig-zag line with three segments. PSPC uses a standard Cartesian representation for pairs (X_1, X_2) and (X_3, X_4) that is familiar to everybody with high-school background. In contrast, the PC need to be learned.

The only novelty that a user needs to learn in Figure 10(a) is a shift of coordinates (X_3, X_4) relative to (X_1, X_2). The coordinates are labeled, thus, it is quite intuitive and dotted lines help to trace values in (X_3, X_4) coordinates that interactive software can provide, if needed.

Figure 10(a) exploits the PSPC property that n-D points with values of coordinates that are similar to the values of coordinates of the anchor n-D point are visualized as smaller graphs (see mathematical statements in Appendix 1). In 4D case, it means smaller arrow as Figure 10(a) shows. This is also a quite intuitive metaphor that cases that are closer to the goal are more similar to the goal in its visual representations.

Next, Figure 10 uses color to indicate the progress in reaching the goal. The initial health status is shown as a red arrow. Then, the arrows that are closer to the goal are shown in yellow and light green with the goal shown as a dark green dot. A few informal experiments that we conducted with participants had shown that people very quickly grasp how to use PSPC for such health monitoring. More formal studies will be conducted later.

While this example used four health indicators, it can be expanded to incorporate more such indicators. For instance, adding two more health indicators will just add another pair of shifted Cartesian Coordinates. The goal still will be a single dark green 2D dot with each graph that represents the status at time-t consisting of two connected arrows. These graphs become smaller when they approach the goal point similar to that shown in Figure 6(b) for 6D points.

Iris data classification in two-layer visual representation

First-layer representation. Iris 4D data with three classes of Iris from UCI Machine Learning repository are common data for evaluating algorithms. Each class contains 50 4D records with total 150 records. Figures

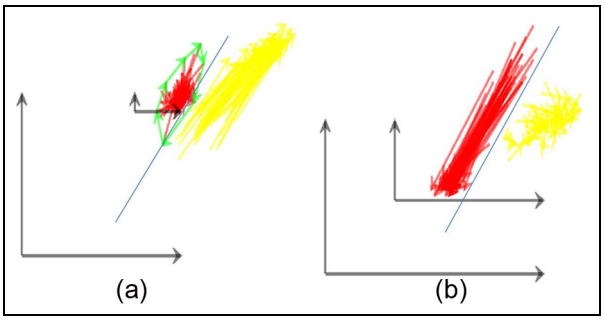

Figure 11. (a) Iris data in Parametric Shifted Paired Coordinates (PSPC) anchored in class 1—the class 1 convex hull is in green and (b) Iris data in PSPC anchored in class 2.

below show the process and results of visual discovery of rules that allow discriminate these classes.

Figure 11(a) shows the results of representation of Iris classes 1 and 2 in PSPC with the anchor point as the middle point of cases of class 1, and Figure 11(b) shows the same two classes with the anchor point as the middle point of cases of class 2.

Middle 4D points are computed as $(\min(x_i) + \max(x_i))/2$ for each attribute in the respective class. In both pictures, classes 1 and 2 are clearly visually separated while the separation is better in (b). The blue lines in both figures are separation lines discovered visually. Those lines can be formalized for analytical linear discrimination of these classes.

In Figure 11, Iris class 1 is shown in red and Iris class 2 is shown in yellow. Figure 11 also shows a convex hull (in green) for Iris class 1. In addition to the black lines, a user can easily construct other discrimination lines visually at the different levels of generalization, that is, how far from the given points each class is extended. These extensions can be non-convex hulls, convex hulls, extended convex hulls, and straight (linear) discrimination lines shown in Figure 11. Comparison of classes in Figure 11(a) and (b) shows that the class that is used as a source of the anchor point in PSPC is visually represented as a smaller and more compact blob. It is in full accordance with PSPC concept and methodology. In Figure 11(a), it is a small red blob, and in Figure 11(b), it is a small yellow blob. The anchor point A is represented as a single 2D point, and 4D points that are close to it are represented as small graphs around it.

Next, we show the separation of classes 2 and 3. Figure 12 shows the graphs of the n-D points of classes 2 and 3 in PSPC with the anchor point as the average point of class 2. The graphs of n-D points of class 2 are in yellow. All graphs of class 3 are in light blue in Figure 12(a). In Figure 12(b), the graphs of

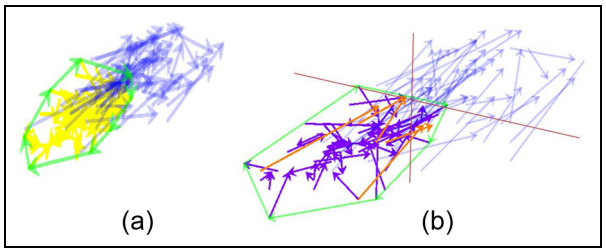

Figure 12. Graphs of n-D points of classes 2 and 3 in PSPC: (a) graphs of 4D points of class 2 (yellow) and class 3 (light blue) with shifted coordinate axis relative to the average point of class 2 used as the anchor point and (b) graphs of 4D points of class 2 (dark blue), class 3 (light blue), and class 3 that are fully within the convex hull of class 2 (orange).

class 3 that are fully within the convex hull of class 2 are in orange. The shifted coordinate axes shown in Figure 12(a) are computed relative to the average point of class 2 used as the anchor point. At first glance, here, classes 2 and 3 heavily overlap. In fact, this is not the case.

Most of the end points (x_3, x_4) of the arrows that represent 4D points of class 3 are on the right of one red line and above another red line in Figure 12(b). Respectively, there is a simple rule that separates class 2 from class 3 based on these two red lines in the coordinates (X_3, X_4). The first line is a vertical line $x_3 = e$, where e is some constant extracted from Figure 12(b). The second line has an equation $d_3x_3 + d_2x_4 + d_{12} = 0$, where the coefficients also extracted from Figure 12(b). Thus, this rule is as follows for a 4D **x** point

$$\text{If } (x_3 > e) \text{ or } (d_3x_3 + d_2x_4 + d_{12} > 0) \text{ then x belongs to class 3} \quad (1)$$

An alternative way to separate the cases of class 3 that are only partially within the convex hull of class 2 is building a rule that directly uses the convex hulls

$$\text{If } (x_1, x_2) \notin H_1 \text{ and } (x_3, x_4) \notin H_2 \text{ then } \mathbf{x} \text{ is in class 3} \quad (2)$$

where (x_1, x_2) is a part of the 4D point $\mathbf{x} = (x_1, x_2, x_3, x_4)$, H_1 and H_2 are convex hulls of classes 1 and 2. The accuracy of both rules for classes 2 and 3 is the same $(50 + 45)/(50 + 50) = 0.95$ with 50 cases in class 2 and 50 cases in class 3 due to the fact that five cases of class 3 are fully within the convex hull of class 2 (see orange cases in Figure 12(b)).

Second-layer representation. The next step is an attempt to improve this accuracy using *the second layer of visual discovery* by

- Extracting visually the features from graphs of class 2 and from misclassified graphs of class 3 in Figure 12 that can potentially discriminate these cases, and then
- Discovering new classification rules visually based on those new secondary features.

While below we demonstrate this approach for Iris data that the concept of the two-layer representation is a part of a general concept of *multilayer visual knowledge discovery*. When classes are not fully separated in the visualization in original features, this first-layer visualization serves as a source of information for extracting features of the second layer to be used to fully separate classes. Similarly, the features of the second layer serve as a source for the third layer and so on if needed.

The visual analysis of Figure 12 shows that orange arrows are closer to the anchor point **a** (in the middle of the dark blue arrows) than other arrows from class 3. This leads to the extraction of the distance between **x** and **a** and the distance between the ends of **x** and the anchor point **a** in coordinates (X_1, X_2). Another subtle feature is associated with the length of orange lines relative to arrows of class 2. Several of them are longer than arrows of class 2. This leads to extracting lengths of arrows. To simplify computations, we computed horizontal and vertical projections of length of arrow. The results of this visual feature extraction written for 4D point **x** and the 4D anchor point $\mathbf{a} = (a_1, a_2, a_3, a_4)$ are as follows:

$y_2 = ((x_3 - a_3)^2 + (x_4 - a_4)^2)^{1/2}$—the distance between points **x** and **a**
$y_1 = ((x_1 - a_1)^2 + (x_2 - a_2)^2)^{1/2}$—the distance between the ends of **x** and the anchor point **a**
$y_3 = x_1 - x_3$—horizontal coordinate difference
$y_4 = x_2 - x_4$—vertical coordinate difference

Figure 13 shows the results of the representation of 50 cases of class 2 and 5 misclassified cases of class 3 in y_1, y_2, y_3, y_4 coordinates in PSPC, with the anchor in the average case of class 2 in y_1, y_2, y_3, y_4 coordinates.

Figure 13 allows visual finding of two areas A and B where only points of class 3 are present. This leads to a rule

$$\text{If } (y_3, y_4) \in A \text{ or } (y_1, y_2) \in B \text{ then class 3} \quad (3)$$

It classifies all five cases of class 3 correctly with total 100% accuracy. Instead of $(y_3, y_4) \in A$, we can use in equation (3) another more general and robust condition $(y_3, y_4) \notin H_2$ and $\mathbf{y} \notin H_1$, where H_1 and H_2 are the convex hulls of classes 1 and 2. The next

Table 2. SVM confusion matrixes for Iris data.

Radial kernel[43]				Radial kernel[42]				Polynomial kernel[42]			
Real class	Predicted class			Real class	Predicted class			Real class	Predicted class		
	1	2	3		1	2	3		1	2	3
1	50	0	0	1	50	0	0	1	50	0	0
2	0	48	2	2	0	47	3	2	0	46	4
3	0	2	48	3	0	2	48	3	0	1	48

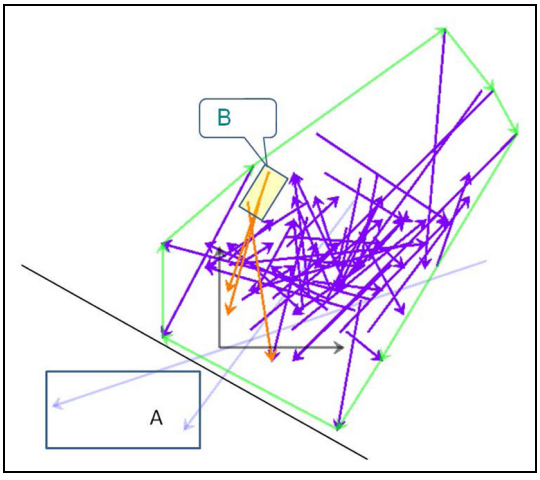

Figure 13. Second-layer visual representation: dark blue—50 cases of class 2, orange and light blue—5 cases of class 3 in y_1, y_2, y_3, y_4 coordinates in PSPC.

robust but less accurate rule for Figure 13 is rule (4) with three cases of class 3 in area B misclassified based on the black line L that separates area A from the green convex hull of class 2

$$\text{If } L(y_3, y_4) > 0 \text{ then class 2, else class 3} \qquad (4)$$

Comparison with PC, Radvis, and SVM. Figure 14 shows all three Iris classes in PC[41] and Radvis.[54] Both clearly show the abilities to separate only class 1 from classes 2 and 3, but do not separate well the classes 2 and 3. Table 2 shows the results of SVM algorithm on the same Iris data.[42,43] In both SVM models from Kaur,[42] the total number of errors is 5, which are in classes 2 and 3.

This is exactly the same number of errors that we obtained in Figures 11 and 12 before the second layer of visual discovery. Results from Mafrur[43] with four errors are slightly better than our result and from Kaur,[42] but it uses 59 support vectors (over 1/3 of all cases) that may indicate overfitting. For class 2, it has 23 support vectors. Our solution uses only nine 4D border points of the convex hull. Our alternative solution with two 2D linear discrimination lines in Figure 12(b) only needs six scalar coefficients with very similar accuracy as in Mafrur.[43]

Note that while our second layer brings 100% accuracy, it may be overfitting for the area B in Figure 13 that is responsible for eliminating three errors. In

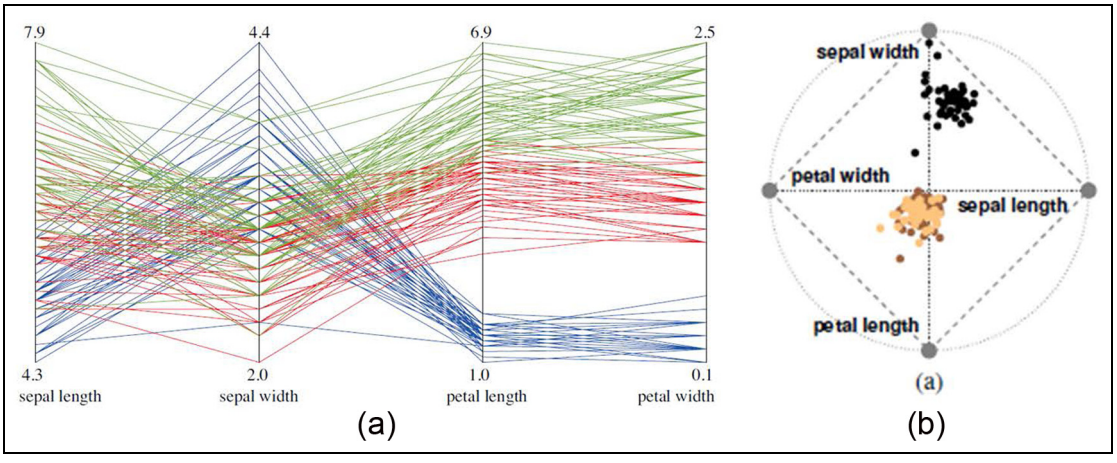

Figure 14. Parallel Coordinates and Radvis visual representation of iris classes: (a) Parallel Coordinates representation[41] and (b) Radvis results (Rubio-Sanchez et al., 2016).

contrast, the area A can be generalized by a linear discrimination line (a black line in Figure 13) that needs just three scalar coefficients. The accuracy of this solution is 98% (147/150). The advantage of visual analysis is that a user can see areas A and B and judge how artificial and complex they are to decide which one to ignore to avoid overfitting.

While this analysis does not involve cross-validation for testing models, the presented GLC visualizations are informative to the possible success or failure of cross-validation. It is visible in Figures 11–13 that the common leave-one-out cross-validation will not significantly change the accuracy produced by the GLC methods.

Next SVM will classify every 4D point as belonging to one of the three classes being trained on them, while Figures 12–14 show the significant areas without any 4D point from the given 150 4D points. Therefore, the refusal to classify 4D points in such areas can be justified. In this case, the points outside of the convex hulls in Figures 12–14 will not be classified into these three classes. When new genetically modified Iris is introduced, the 4D points in these areas can get their classification, but to another class not to one of these three classes.

Experimental comparison of visual recognition of 160D linear data structures with Gaussian noise in PC, Stars, and CPC Stars

Section "Advantages of GLC for real data visualization" focused on developing and applying new GLC methods on 4D real-world data. Below we explore GLC in 160D on simulated (modeled) data in user studies without involving methods presented in section "Advantages of GLC for real data visualization." This section experimentally explores the human abilities to recognize 160D linear patters with Gaussian noise. Those patterns are represented by n-D points within linear hypertubes (hypercylinders) in 160D where each hypertube represents a data class. The axis of the hypertubes are given as $A_k + t\mathbf{u}_k$, where A_k is a starting n-D point of the axis, $t \in [0,1]$ and n-D vector $\mathbf{u}_k$ sets up a direction of the hypertube. The noise level defines the width of each linear hypertube.

The essence of this study is the visual shape recognition by humans. It is drastically flexible and a very complicated process. Many years of psychology development produced very general *qualitative* Gestalt laws, but only very limited *quantitative* estimates of perception thresholds for simple shape features such as line length, box size, or brightness. The reasons of this limited progress are as follows: (1) *mutual influence* of many features in recognition of complex form and (2) *deficiencies of modeling theories* for such complex processes. As a result, building of quantitative applied model of vision is extremely difficult and time-consuming.

Therefore, this study focuses on experimental *qualitative ranking* test on what displays are essentially better than others. Our experiment on a modeled data structure allows extending its results only *qualitatively* onto data with similar structures. The actual numbers from such tests are very dependent on many factors (objective and subjective).

This dependence and multiple external assumptions of the statistical theory limit applicability of the statistical theory and criteria to such type of data.[44–46] These limitations led the journal of Basic and Applied Social Psychology (BASP) to banning the null hypothesis significance testing procedures (NHSTP) and estimation of confidence intervals:[44] "Analogous to how the NHSTP fails to provide the probability of the null hypothesis, ... confidence intervals do not provide a strong case for concluding that the population parameter of interest is likely to be within the stated interval."[44] This journal keeps basic descriptive statistics which we do below estimating means and standard deviations.

Experiment goal and setting

The goals of the experiment are to

1. Test *effectiveness* of some GLC for visual discovery of n-D data structures for data with over a 100 dimensions;
2. Identify *advantages* of Radial Coordinates (traditional Stars) and Star CPC;
3. Further expose the advantages of *modeled data approach* of visual analytics, as allowing results generalization versus getting results applicable only to very specific analyzed data.

First we outline *the modeled data approach* to data generation for the experiment that was formulated by Grishin and Kovalerchuk.[13] Testing new visualization methods is possible in two fundamentally different ways. The first one is generating data with *given mathematical properties*. When the method is successful in experiments on these modeled data, this method can be successfully applied to another data with the same mathematical properties. The second way is to experiment with data without known mathematical properties, which is common in uncontrolled real-world data. In the last case, the success of visual representation of such data does not help to know how successful this method can be on another data, because the properties

of data were not formulated. Thus, the judgment about the method effectiveness is quite limited under such testing. Only if the solution for these specific data has its own value, beyond the illustration of the method success, then such tests are beneficial by themselves, but not for other datasets. However, not all real data are such "self-beneficial" data; therefore, a wide use of modeled data will be beneficial for testing many other new methods, not only GLC.

Data modeling steps implemented in this experiment include the following:

Step 1. Randomly generating the linear hypertubes (hypercylinders) that cross the origin of the n-D data space of dimension n = 160. All hypercylinders are normalized to length 1 with the axis A + t**u**, where A is its starting, $t \in [0,1]$, and n-D vector **u** is a hypertube direction. Both A and **u** are randomly generated.
Step 2. Computing randomly equidistant points on axis of these hypertubes in the range from t = 0.3 to t = 1.0, that is, forming a set of vectors $\{\mathbf{v}_i\} = \{v_{i1}, v_{i2}, ..., v_{in}\}$ from the origin to these points.
Step 3. Computing n-D vector $\mathbf{k}_G = (k_{G1}, k_{G2}, ..., k_{Gn})$ of Gaussian noise with standard deviation $\sigma \in [0.1,0.3]$ for each $\mathbf{v}_i$ separately.
Step 4. Computing n-D points $\mathbf{w}_i = (k_{G1}v_{i1}, k_{G2}v_{i2}, ..., k_{Gn}v_{in})$, that is, vectors $\mathbf{v}_i$ with multiplicative noise.

Data visualization steps implemented in this experiment include the following:

Step 1. Selecting visualization method M_k: Regular Stars, CPC Stars, and PC.
Step 2. Displaying each generated n-D point $\mathbf{w}_i$ using M_k in a separate window.
Step 3. Tiling these windows in the random order (see Figures 15–17).
Step 4. Repeating steps 1–3 for other selected visualization methods M_k.

Task and solving hints

In the actual experiment, the participants observed 160D data. In the introduction session, participants observed other data and of a smaller dimension

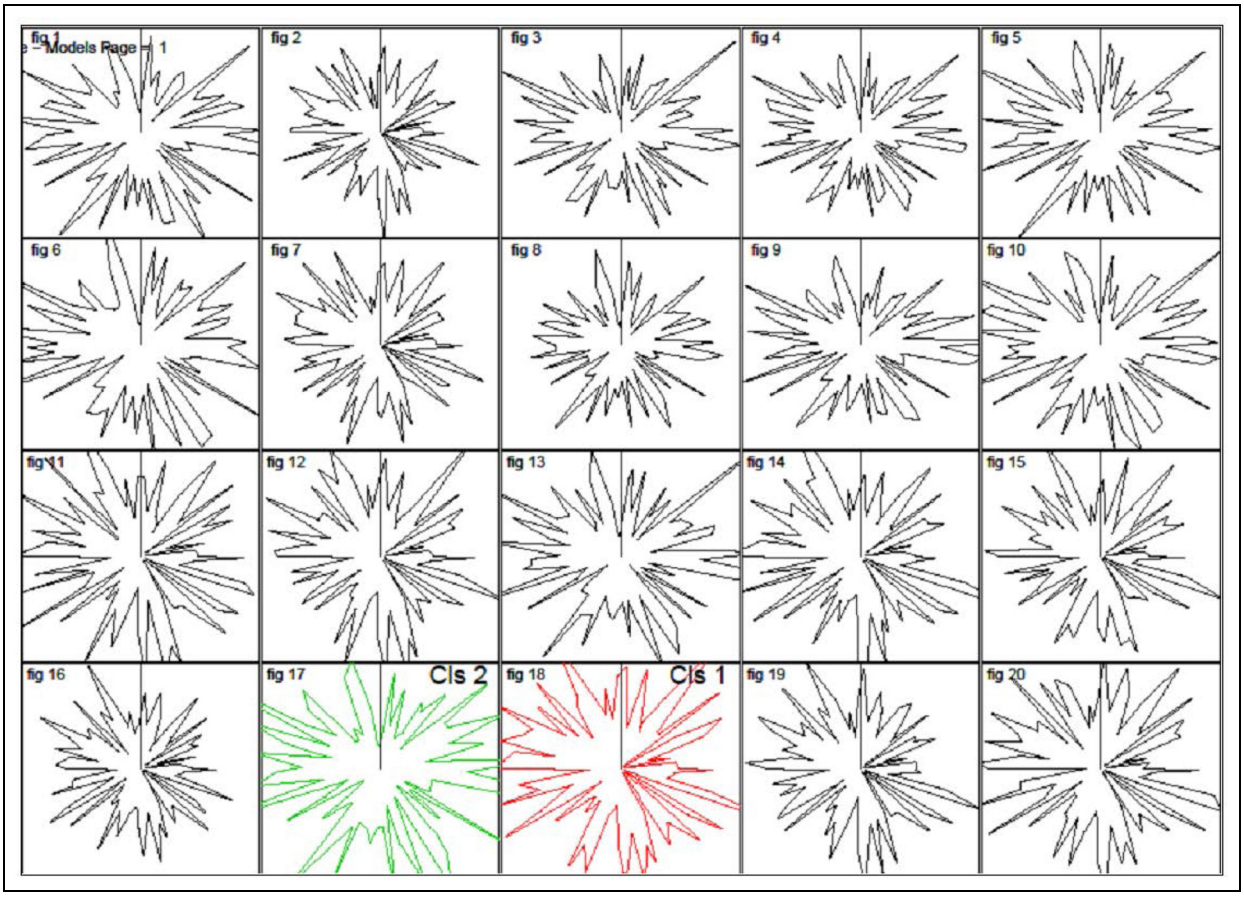

Figure 15. Twenty 160D points of two classes represented in Star CPC with noise 10% of maximum value of normalized coordinates (max = 1) and with standard deviation 20% of each normalized coordinate.

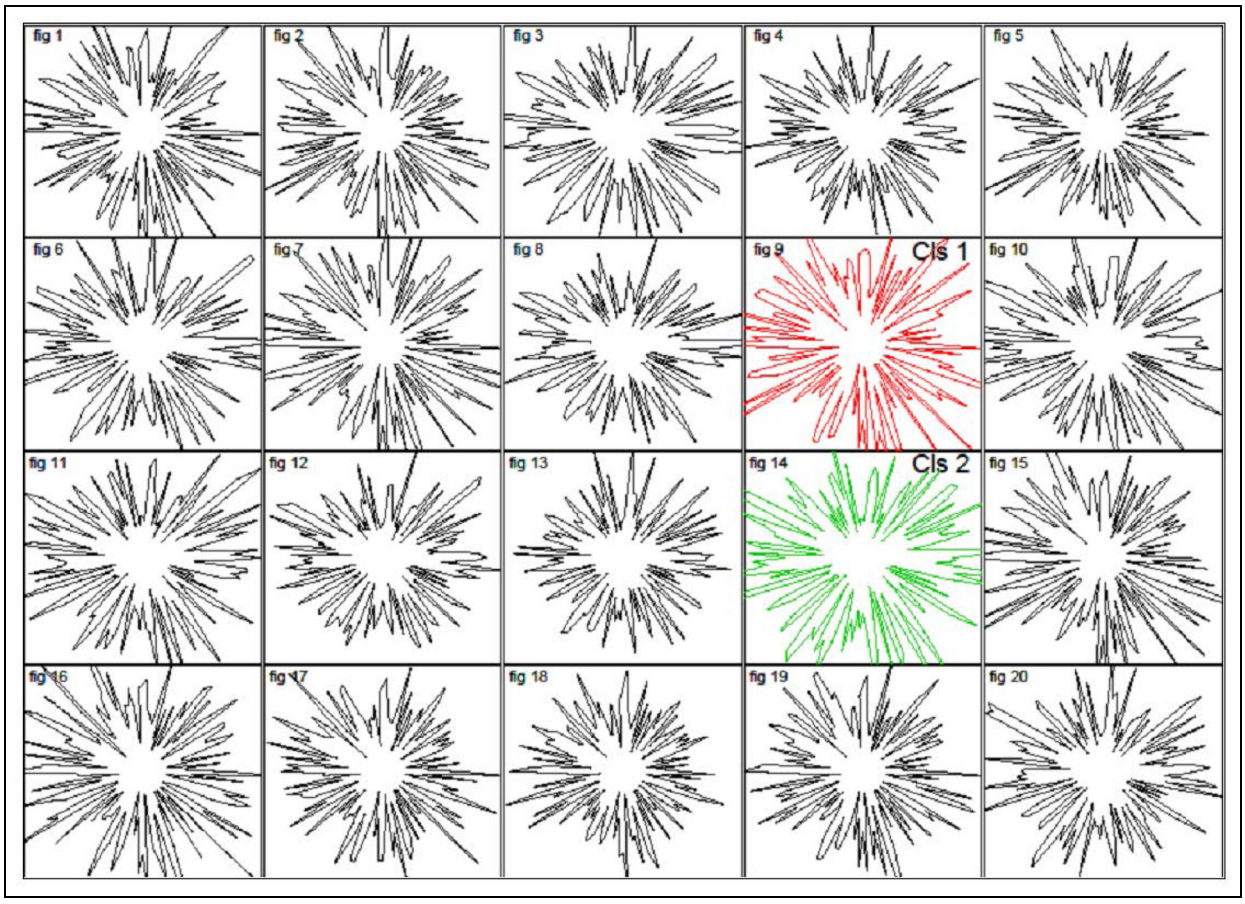

Figure 16. Twenty 160D points of two classes represented in Radial Coordinates with noise 10% of maximum value of normalized coordinates (max = 1) and with standard deviation 20% of each normalized coordinate.

n = 100. The goal of introduction session was to make participants familiar with the experiment setup not data of actual experiment. The actual 160D training data were provided to participants only during the experiment as two labeled figures on the same sheets of paper where the 18 test cases were present (see Figures 15–17). Each participant received the three sheets of paper in A4 format with 20 figures in each sheet generated by a given method as shown in Figures 15–17. Radial Coordinates, Star CPC, and PC methods are used in these sheets to visualize n-D points. These 20 figures split equally between two classes. Participants are informed about equal number of figures of two classes, but only one figure of each class is labeled by the class number and distinctly colored. These two figures serve as training data. The locations of figures of the same class in the three sheets are randomized to eliminate the impact of location on results of the experiment.

The goal of the participant in the actual experiment is to classify unlabeled 18 figures using two labeled training figures within 20 min per sheet (1 h total for three sheets). In the pilot study, we found that it is not required more than an hour for the images of this complexity. A participant is asked to write the class number next to each unlabeled figure. It was also recommended to participants to circle up to 3–5 found local patterns in unlabeled figures that match the training figures. Each participant worked with total $18 \times 3 = 54$ stimuli to be recognized. These stimuli are presented in three sheets shown in Figures 15–17.

In the introduction session, with 100D data to help participants to better understand examples of patterns of radial directions, angles, convexities, concavities, different forks, figure symmetries, envelops, orientations of parts, elongations, and so on distorted and not distorted by noise have been provided. Figure 18 shows the samples of these data. This was a part of the lecture delivered to participants (all participants are students majoring in Computer Science). The lecture explains to them the design of all three visualization methods with several examples.

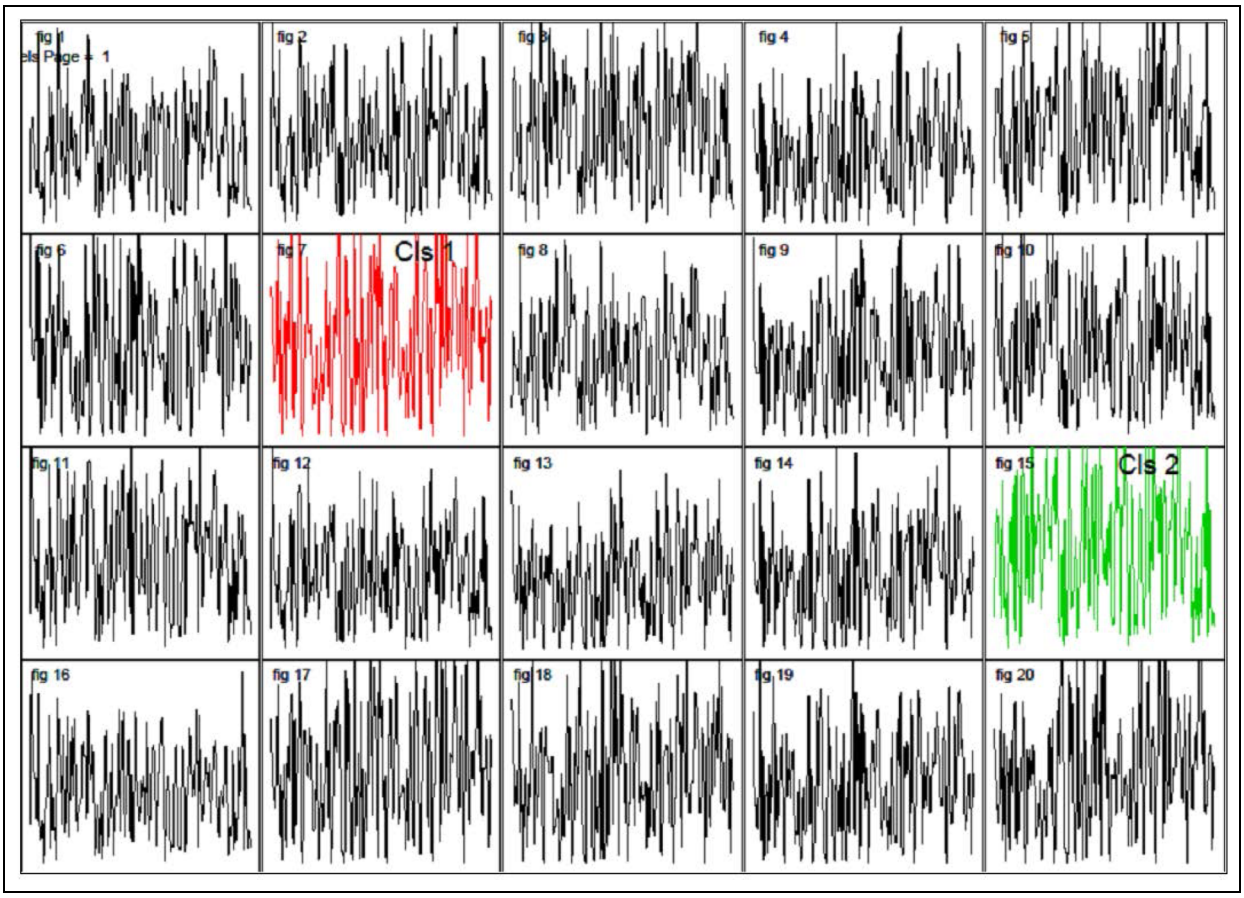

Figure 17. Twenty 160D points of two classes represented in Parallel Coordinates with noise 10% of maximum value of normalized coordinates (max = 1) and with standard deviation 20% of each normalized coordinate.

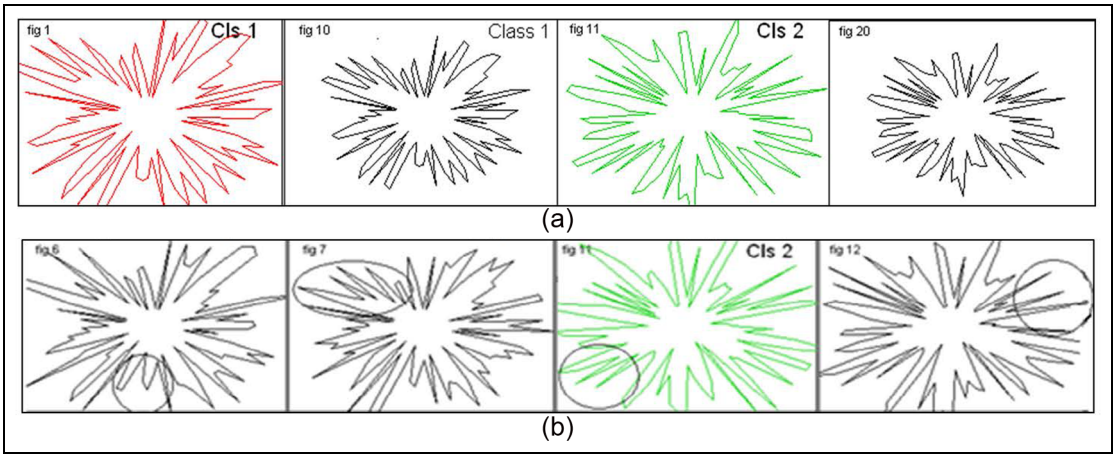

Figure 18. Samples of 100D data in Star CPC used to make participants familiar with the task: (a) initial 100D points without noise for class 1 (hypertube) and class 2 (hypertube) and (b) 100D points with multiplicative noise: circled areas are the same as in upper star.

Table 3. Results of the experiment with 160D data based on answer by 60 respondents who filled all forms on 160D data classification.

	PC				Stars				CPC Stars			
	Errors	Refusals	Errors + refusals	Correct	Errors	Refusals	Errors + refusal	Correct	Errors	Refusals	Errors + refusal	Correct
Answers	189	35	224	856	78	10	88	992	59	9	68	1012
Mean per person	3.15	0.58	3.73	14.27	1.30	0.17	1.47	16.53	0.98	0.15	1.13	16.87
Mean %	17.50	3.24	20.74	79.26	7.22	0.93	8.15	91.85	5.46	0.83	6.30	93.70
SD	2.52	1.44	2.42	3.03	1.87	0.46	1.57	1.9	1.8	1.22	1.77	1.88

Results and comparison with previous experiment

The experiment was conducted with two groups of computer science students at two universities. Over 100 sets of forms were distributed in one university and 15 sets in another one. In total, 75 sets were returned from the first set and 14 sets were returned from the second set. Not all forms were fully filled. Table 3 shows the results of this experiment based on these responses.

In total, 18 students did not make any errors and 6 students made one error in all tests. It shows that there is a room to increase both noise level and dimensions. It also shows that to reveal differences in difficulties in these three displays (PC, Stars, and CPC) for these students, the noise level and/or dimension has to be increased in further tests. On the other side, three students could not solve all three tests at all. One student made Stars and CPC Stars tests without errors, but could not make PC test, and one student could not solve two tests. The actual reasons are not known while possible reasons can be rushing, lack of motivation, insufficient test time, and time to become familiar with test setting, and other personal reasons.

A total of 60 students provided *legible answers in all* PC, Stars, and CPC displays but some figures may left unanswered. The figures left without answers were interpreted as refusals, and incorrect class labeling was interpreted as an error. The results of these 60 students are shown in Table 3. The results in Table 3 and Figure 19 (that visualizes some data from Table 3) show that

- Respondents are able to find multiple noisy patterns in 160D data presented in all three visualization methods in a short period of time (within 1 h).
- Classification in PC was three times less accurate that in Radial Coordinates and Star CPC (224 vs 88 and 68 errors and refusals).

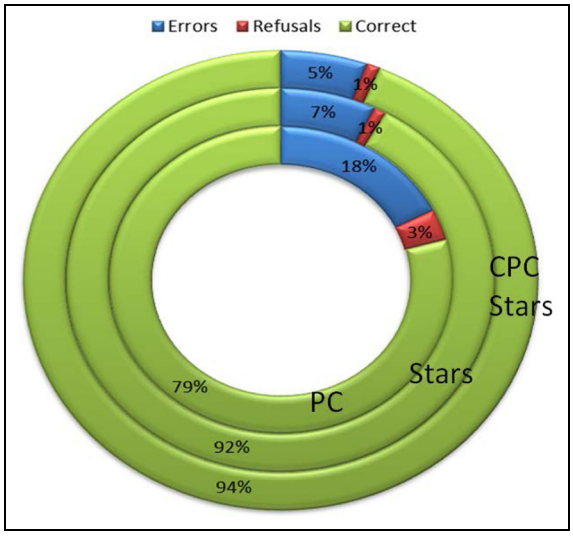

Figure 19. Comparison of results of the experiment on PC, Stars, and CPC Stars.

- Classification in Radial Coordinates was slightly less accurate (14%) than in Star CPC (total 88 vs 68 errors and refusals), but many students solved CPC Star tests faster than Stars.
- The number of refusals in PC was 2–3 times greater than in Radial Coordinates and Star CPC (35 vs 10 and 9 refusals).

In an informal interview after the experiment, a number of respondents stated that classifying figures and finding patterns in Star CPC were *easier* than in Radial Coordinates and in both were easier than in PC. Respondents also stated that they have *more confidence in their decision in Star CPC than in other methods due to less complexity* of the figures. Only the nine refusals in Star CPC classifications confirm such informal statements.

Below we compare the results of this experiment with an experiment in Grishin and Kovalerchuk[13] with

much less number of respondents, but with much more difficult data of dimension n = 192. In that experiment, respondents were asked to find a few features separating each of five tubes (classes). Each class was represented by using CPC Stars randomly placed on a sheet of paper for six 192D points. Also, the respondents solved the same task in Radial Coordinates representation of the same n-D data points. The experiment was repeated for different levels of "noise," that is, hypertube width (up to 10%, 20%, and 30% of maximum possible value of each coordinate of an n-D point), with time limit of 20 min allowed for each GLC representation type.

That experiment had shown that with increased noise level from standard deviation, 10% to 30% respondents not only produced more misclassifications but one of the respondents completely refused answering using Radial Coordinates for data with 20% and 30% of noise and another one with advanced skills refused answering using Radial Coordinates for data with 30% of noise. In contrast, CPC Star tests showed acceptable time (1–5 min) for all figures with noise up to 30%. The success of CPC Stars versus Radial Coordinates was due to two times less dense placement of the points as a result of a specific mapping of pairs (x_i, x_{i+1}) into a closed contour in CPC Stars.

The difference in two experiments is not only in the dimensions and the number of classes (2 vs 5) but also in the noise control. In the current experiment, the Gaussian noise is with mean 10% of the maximum value of each normalized coordinate (max = 1) and standard deviation of 20% of that maximum value for the normal distribution, N(0.1, 0.2), that is, this noise is three times smaller than the highest noise of 30% used in Grishin and Kovalerchuk,[13] which is much more complex for the human analysis.

The previous experiment with n = 192 and a high level of noise (30%) point out that likely the *upper bound* of human classification of n-D data using the Radial Coordinates for data modeled as linear hypertubes is no greater than 192 dimensions with up to 30% noise. One of the motivations for the current experiment with n = 160 was the failure of Radial Coordinates at n = 192. We decreased dimensions and noise level to find out would n = 160 with lower noise be upper bound or not for the Radial Coordinates. The current experiment shows that the upper bound for human classification on such n-D data is not less than n = 160 dimensions with up to 20% noise. Thus, the expected *classifiable dimensions are in* [160, 192] *dimensions interval* for the Radial Coordinates.

Due to advantages of Star CPC over Radial Coordinates, these limits must be higher for them and lower for PC due to higher occlusion in them. More exact limits are the subject of the future experiments. The current experiment was conducted with about 70 respondents, thus it seems that 160 dimensions can be viewed as a quite firm bound. In contrast, the question that 192 dimensions is the maximum of the upper limit for Star CPC may need additional studies. Thus, so far, the indications are that the upper limit for Star CPC is above n = 192 and it needs to be found in future experiments for linear hypertubes. Note that finding bounds for linear hypertubes most likely will be also limits for non-linear hypertubes due to their higher complexity.

Visual representation of n-D relations in GLC

In Paired GLC such as CPC and SPC, each directed edge of the graph directly visualizes relations of four dimensions (section "Background definitions"). In contrast, each edge of the graph in PC directly visualized only relations between two adjacent coordinates. For instance, for all coordinates scaled to [0.1], in 4D CPC, an edge going up and to the right indicates relation $(x_3 > x_1)$ and $(x_4 > x_2)$. In contrast, in PC, the edge going to the same direction (up and right) indicates only relation $(x_2 > x_1)$. In CPC, if this edge is in the first quadrant below its diagonal, then in addition, it also expresses the relation $(x_2 < x_1)$ and $(x_4 < x_3)$. This section continues exploring relations in n-D spaces. It includes visualization of 2D linear dependencies in PC, CPC, and SPC (section "2D linear dependencies in PC, CPC, and SPC"), n-D linear dependencies in PC and CPC (section "n-D linear dependencies in PC, CPC, and SPC"), and linear dependencies in GLC for classification and numerical estimates (section "Linear dependencies in GLC for classification and numerical estimates").

2D linear dependencies in PC, CPC, and SPC

In PC, a 2D line L: $x_j = mx_i + b$ is visualized by an *infinite set of lines*[3] (Figure 20(a) and (c)). In CPC and SPC, the same line is represented in a classical Cartesian form as a *single line* (Figure 20(b) and (d)), because CPC and SPC consist of a set of pairs of classical Cartesian Coordinates that are collocated or shifted. In PC, there is a point L* that does not show the value of x_j having x_i. One must draw a line to coordinate X_j via points x_i and L* to get x_j.

For each other x_i value, a different line must be drawn. If all these lines are drawn together, they will completely cover a large segment between coordinates X_i and X_j and none of the line will be visible creating an extreme case of full occlusion (see the gray area in Figure 20(a)). In contrast, in the classical Cartesian

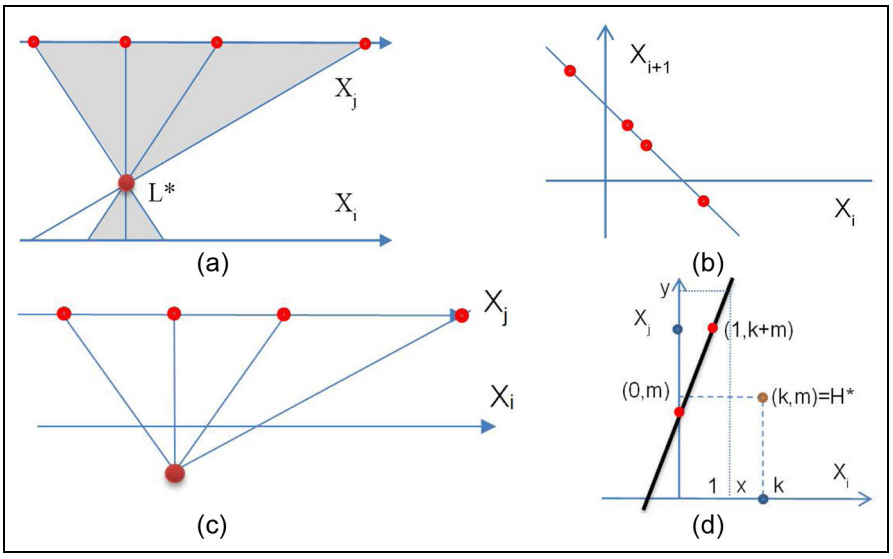

Figure 20. Lines $y = mx + b$ in (a, c) Parallel Coordinates and (b, d) Collocated and Shifted Paired Coordinates; (a) line L: $x_j = mx_i + b$ for $m < 0$ and point L* that represents L in Parallel Coordinates, (b) line L: $x_j = mx_i + b$ for $m < 0$ in Cartesian visualization used in CPC and SPC for pairs of coordinates (X_i, X_{i+1}), (c) line L: $x_j = mx_i + b$ for $m > 0$ and point L* that represents L in Parallel Coordinates, and (d) line L: $x_j = mx_i + b$ for $m > 0$ in Cartesian visualization used in CPC and SPC for pairs of coordinates (X_i, X_{i+1}) with point H* that completely represents L.

Coordinates, the same single line is used for all x_1. This classical metaphor is familiar to everyone and learning a new metaphor is not required. Also, while point L* fully represents line L: $y = kx + m$, the 2D point $H^* = (k, m)$ fully represents L also in a compact way in CPC in a pair of coordinates such as (X_1, X_2). The visual process of getting y from a given x is drawing line L using points (0, m) and (1, m-k), then drawing a line from x on X_1 to L, and projecting the crossing point to X_2 to get y (see Figure 20(d))

In SPC, we have two cases for representing line L. The first case is for consecutive pairs such as (X_1, X_2), and (X_3, X_4), where L is visualized in the classical Cartesian form discussed above. The second case is for odd pairs such as (X_1, X_3), (X_3, X_5), and (X_5, X_7), where L: $x_j = mx_i + b$, $j = i + 2$. In SPC, these coordinates are parallel but shifted, thus the classical Cartesian visualization is not applicable. However, this situation is the same as for shifted PC considered in section "Decreasing occlusion by shifting coordinates." When X_j coordinate is shifted, the point L* can keep its locations in shifted PC and SPC if one or both coordinates are rescaled to accommodate the shift. Alternatively, L* is shifted if coordinates are not rescaled. To see, it is sufficient to compare Figures 20(a) with Figure 21(a) where X_j is shifted. Thus, the property that L* fully represents a 2D line L in PC holds for both SPC and Shifted PC.

Now consider situations for CPC. For consecutive pairs (X_i, X_{i+1}) for odd i, it is the same as for SPC considered above, but for collocated pairs (X_i, X_{i+2}) with a common origin such as (X_1, X_3) and (X_2, X_4), the situation is different and another solution is required. This solution is shown in Figure 21(b)–(d). First, a distance d is set up and coordinate X_{i+2} is moved up to this distance parallel to X_i. In this way, we get a situation of finding point L* as it is done in PC in Figure 20(a). This stage is shown in Figure 21(b), where blue dotted lines show how x_j is identified on X_j by getting line via x_i and L*. Next, these lines are reflected back to X_i relative to the middle line that is at height d/2 for a given d (see Figure 21(b)) to get value of x_j on coordinate X_j that is collocated with coordinate X_i. This process is equivalent to building a triangle with two equal sides. Thus, the X_j can be removed from its temporary location (see Figure 21(c)). In the final algorithm, the triangular property allows avoiding moving X_j to the temporary location substituting it by building triangles. Finally, Figure 21(d) shows multiple points L* for 8D data for pairs (X_1, X_3), (X_3, X_5), and (X_5, X_7) in CPC.

n-D linear dependencies in PC, CPC, and SPC

This section explores the representation of n-D linear dependencies in PC, CPC, and SPC. The PC visualization in Figure 20(a) and (c) is not actual visualization of the n-D straight line: $A + t\mathbf{v}$, where A is an n-D point, $\mathbf{v}$ is an n-D vector, and scalar t changes in some interval. It is the visualization of a single *projection* of

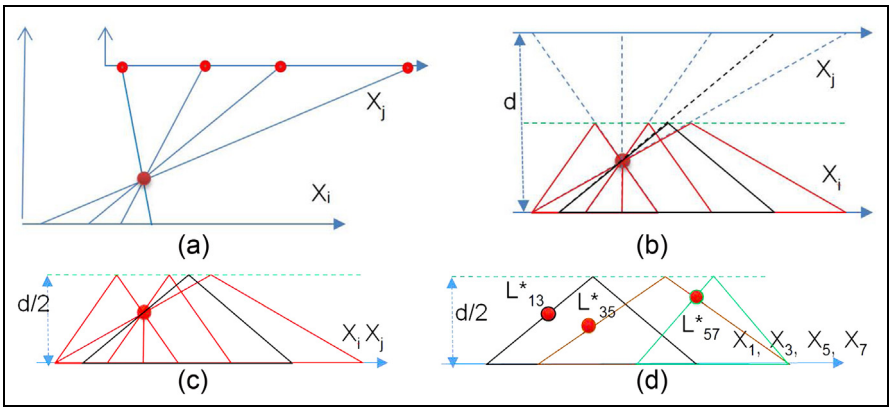

Figure 21. Line L in shifted PC, CPC, and SPC; (a) line L: $x_j = mx_i + b$ and point L* that represents L in Shifted Parallel Coordinates, and SPC where both i and j are odd or even; (b) line L: $x_j = mx_i + b$ in CPC constructed by reflecting points relative to the middle line at height d/2 for a given d; (c) Figure (b) with removed X_j from its temporary location; and (d) multiple points L* for 8D data for pairs (X_1, X_3), (X_3, X_5), and (X_5, X_7) in CPC.

that n-D line to a pair of coordinates (X_i, X_j).[3] To fully represent the n-D straight line, A + t**v**, Inselberg[3] uses *a set of projection* of that n-D line to a pair of coordinates (X_i, X_2), (X_2, X_3), ..., (X_{n-1}, X_n) using the property that a line in R^n is the intersection of n−1 non-parallel hyperplanes. Those hyperplanes can be built from those projections, for example, $x_2 = mx_1 + b$ can be converted to the n-D hyperplane $x_2 = mx_1 + b + 0x_3 + 0x_4 + \cdots + 0x_n$, that is, any n-D point $\mathbf{x} = (x_1, x_2, x_3, x_4, ..., x_n)$ such that $x_2 = mx_1 + b$ will be on that hyperplane when all other x_i for $i > 2$ can take any values due to their zero coefficients.

Conceptually, the idea of this visualization is related to the idea of the scatter plot matrix that represents a set of n-D data in their projections to all pairs of coordinates (X_i, X_j). PC present projections only for adjacent pairs of coordinates, while it is sufficient for restoration of the n-D line it does not show the n-D line itself. This situation is also similar to showing 2D projections of a 3D object instead of this 3D object.

To represent several parallel lines $\{L_k: x_{i+1} = mx_i + b_k\}$ for a given pair (X_i, X_{i+1}) in PC, a set of points L^*_k is used that are located on the same vertical line.[3] In CPC and SPC, the parallel lines are represented by a set of H^*_i points also located on the same vertical line. Alternatively, CPC and SPC have a classical and familiar visual representation of one line shifted above or below another one. As with a single line L, an attempt in PC to draw all points of two parallel lines L_{k1} and L_{k2} will end up with a completely covered "black" segment between coordinates X_1 and X_2. The generalization of this visualization for n-D line, L^n: A + t**v**, in PC requires to show sets of L* points for all consecutive pairs (X_i, X_{i+1}).[3]

This situation with parallel n-D lines also represents another type of linear relations in n-D, where n-D line L^n_2 is a linear function of another n-D line L^n_1 : $L^n_2 = L^n_2 + u\mathbf{h}$, where **h** is a given n-D vector and u is a scalar coefficient. This is equivalent to defining a set of *linear operators* $T_u(\mathbf{x})$ in *n-D space* for different u. At the level of individual $\mathbf{x} = (x_1, x_2, ..., x_n)$, we have $L^n_2(\mathbf{x}) = T_u(\mathbf{x}) = L^n_1(\mathbf{x}) + u\mathbf{h}$, where **h** represents a shift vector of the n-D line. In the notation where $L^n_1 = A_1 + t\mathbf{v}$ and $L^n_2 = A_2 + t\mathbf{v}$, we have $L^n_2 = A_2 + t\mathbf{v} = A_1 + t\mathbf{v} + u\mathbf{h} = (A_1 + u\mathbf{h}) + t\mathbf{v}$. Thus, $A_2 = (A_1 + u\mathbf{h})$, that is, n-D point A_1 moves to direction **h** for the distance $|u\mathbf{h}|$. In other words, the lines L^n_1 and L^n_2 go from points A_1 and A_2 in the same direction given by n-D vector **v**. Under this linear operator for given $\mathbf{x} = A + t\mathbf{v}$, t, $\mathbf{h} = (h_1, h_2, ..., h_n)$ and u, a graph for **x** in PC moves to $(a_1 + uh_1 + tv_1, a_2 + uh_2 + tv_1, ..., a_n + uh_n + tv_n)$. If in vector **v** all its coordinates v_i are equal, then visually in PC, it means that graph for **x** is shifted by that value. This is the case in Figure 22(b). In Figure 22, the 8D structure consists of three 8D points. In CPC, the first 8D point forms a square of four 2D points in 2D (see Figure 22(a)). The 8D linear transform **v** creates four 2D vectors (v_1, v_2), (v_3, v_4), (v_5, v_6), and (v_7, v_8) that transform these for 2D points in different directions depending on the values of these vectors. These vectors are shown in red in Figure 22(a) when all four 2D transformation vectors are the same. In this case, the squares are shifted in the direction of these vectors (see Figure 22(a)). A more complex situation when transformation vectors have different norms but the same direction will deform these squares. Rescaling of coordinates in CPC and SPC allow to make all 2D pairs (v_1, v_2), $(v_3,$

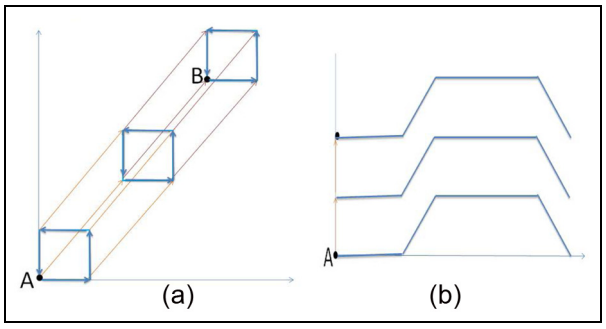

Figure 22. Three linear 8D points in CPC and PC starting from A = (0, 0, 1, 0, 1, 1, 0, 1); (a) three linear 8D points in CPC and (b) three linear 8D points in PC.

v_4), (v_5, v_6), and (v_7, v_8) equal resulting in Figure 22(a), that is, in shifting a graph of n-D point.

Below we consider a case when the n-D lines are corrupted by noise given by vector **e**: $L_2^n = A_2 + t\mathbf{v} + \mathbf{e}$ for **v** with all equal coordinates v_i. We compare visualizations of these lines in PC and CPC in Figure 23.

The method for generation of lines is as follows. First an n-D point, A is generated with a single random number a, A = (a, a, ..., a) for each of 23 dimensions. Then, for generation of all other data points $\mathbf{a}_k$, we add random shift s to all dimensions and a small amount of random noise **e** to each dimension: a_{ki} = a + s + RAND()*5–2.5. In Kovalerchuk and Grishin,[14] the experiment was conducted for n = 23. PC indicated linear correlation as all lines were roughly horizontal (see Figure 23(a)). CPC show those PC lines as small blobs located on a straight diagonal line (see Figure 23(b)) reminiscent to the traditional 2D correlation lines which is a well-known visual metaphor. With less resolution, those blobs are visible as single points strengthening this analogy. Next, these blobs occupy much less space than PC lines. Thus, it is scalable to a larger number of n-D points while PC representation will completely cover PC area where the visual pattern of correlated lines will disappear.

Linear dependencies in GLC for classification and numerical estimates

In the previous section, we considered n-D linear relation where the output is an n-D point not a scalar. This section shows a way to visualize n-D linear function F(**x**) = y where y is a scalar, $y = c_1 x_1 + c_2x_2 + c_3x_3 + ... + c_nx_n + c_{n+1}$. These functions are important in both classification and regression tasks. In regression, F(**x**) directly serves as a regression function, and in classification, F(**x**) serves as a discriminant function to separate two classes with a rule and some threshold T: if $y < T$, then **x** belongs to class 1, else **x** belongs to class 2. The *visualization algorithm* for a linear function F(**x**) = y that we call *GLC- L* consists of steps shown in Table 4.

This algorithm uses the property that cos(arccos k) = k for $k \in [-1, 1]$, that is, projection of vectors $\mathbf{x}_i$ to axis U will be k_ix_i and with consecutive location of vectors $\mathbf{x}_i$, the projection from the end of the last vector $\mathbf{x}_n$ gives a sum $k_1x_1 + k_2x_2 + \cdots + k_nx_n$ on axis U. It does not include k_{n+1}. To add k_{n+1}, it is sufficient to shift the start point of $\mathbf{x}_1$ on axis U (in Figure 24) by k_{n+1}. Alternatively, for the visual classification task, k_{n+1} can be omitted by changing a threshold T to $T - k_{n+1}$ (Figure 25).

Comparison of GLC with Stick Figures (SF)

An SF of n lines ("sticks") connected with different angles encodes 2n attributes.[47] It is done by encoding each pair of attributes (X_i, X_{i+1}) by the length and the angle of the stick (see Figure 26(a)). An SF can look like a human body skeleton, which is a familiar metaphor. Other forms of SFs may not have this familiar metaphor. SFs are useful when figures are shown side-by-side. Otherwise, the occlusion severely limits discovering patterns visually. While GLC also suffer from

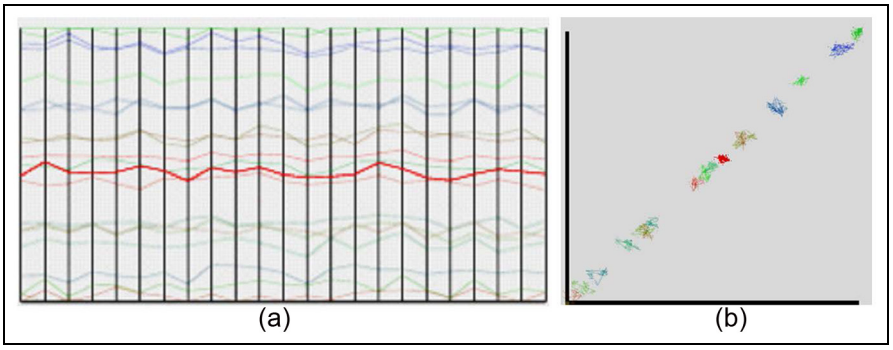

Figure 23. Correlated simulated 23D data in PC and CPC:[14] (a) dataset in Parallel Coordinates and (b) dataset in CPC.

Table 4. Steps of GLC-L algorithm.

Step 1	*Normalize parameters* C = (c_1, c_2, ..., c_{n+1}) by creating as set of normalized parameters K = {k_i}: $k_i = c_i/c_{max}$, where $c_{max} = \|max_{i=1:n+1}(c_i)\|$. The resulting normalized equation $y = k_1x_1 + k_2x_2 + \cdots + k_nx_n + k_{n+1}$ has all $k_i \leqslant 1$ with normalized rule: if $y_{pn} < T/c_{max}$, then **x** belongs to class 1, else **x** belongs to class 2, where y_{pn} is a predicted normalized value by F(**x**). Note that if classes are linearly separated with coefficients C, then these classes are linearly separated with coefficients K, because $y_{pn} = y/c_{max}$. For regression, similarly, we deal with all data normalized, for example, if actual y_{act} is known, then it is normalized too, y_{act}/c_{max} for comparing with normalized prediction.
Step 2	*Compute all angles* $Q_i = \arccos(\|k_i\|)$ of absolute values of k_i and locate coordinates X_1-X_n in accordance with these angles as shown in Figure 24 relative to the horizontal lines. If $k_i < 0$, then coordinate X_i is oriented to the left, otherwise X_i is oriented to the right (see Figure 24). For a given n-D point **x** = (x_1, x_2, ..., x_n), *draw* its values *as vectors* $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ in respective coordinates X_1-X_n (see Figure 24).
Step 3	Draw vectors $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ one after another as shown on the left side of Figure 24. This drawing is in accordance with algorithm 3 (see Appendix 1 for GLC-SC graph constructing algorithm). Then, *project* the last point for $\mathbf{x}_n$ to the horizontal axis U (see a red dotted line in Figure 24). To simplify visualization axis, U can be collocated with horizontal lines that define angles Q_i as shown in Figure 25.
Step 4	*Step 4a*. For regression task, repeat step 3 for all n-D points as shown in the upper part of Figure 25(a) and (b). *Step 4b*. For two-class classification task, repeat step 3 for all n-D points of classes 1 and 2 drawn in different colors. Move points of class 2 by mirroring them to the bottom with axis U doubled as shown in Figure 25. For more than two classes, Figure 24 is created for each class and m parallel axis U_j are generated next to each other similar to Figure 25. Each axis U_j corresponds to a given class j, where m is the number of classes.

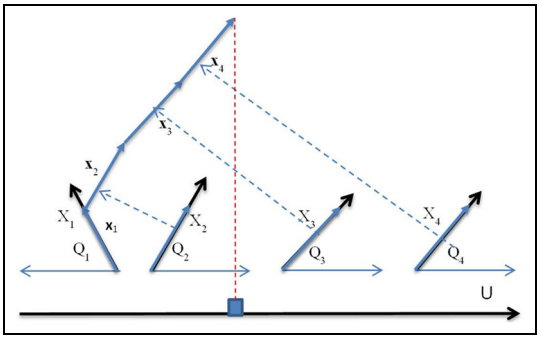

Figure 24. 4D point A = (1, 1, 1, 1) in GLC-L coordinates X_1-X_4 with angles (Q_1, Q_2, Q_3, Q_4) with vectors $\mathbf{x}_i$ shifted to be connected one after another and the end of last vector projected to the black line. X_1 is directed to the left due to negative $k_1 = -0.3$.

occlusion, many GLC including CPC and SPC allow discovering patterns when multiple n-D points are drawn in the same coordinates in a single display as it is shown in Figures 11 and 12 and Kovalerchuk.[12,16] As any glyph approach SFs can be combined with Cartesian Coordinates. In Grinstein et al.,[48] income and age are used to identify the locations of multiple small SFs that create a "texture."

SFs are similar conceptually to Chernoff Faces (CF) that have been compared with GLC in Kovalerchuk.[12] A significant part of this comparison is applicable for comparison of SF and GLC. The major difference of paired GLC from CF and SF is mapping data attributes to visual features. In paired GLC, two attributes are encoded by a single 2D point (a *node* of the graph). In SF, two attributes are encoded by an *edge* ("stick") of the graph (length and angle of the edge). In CF, two or more attributes are encoded by features of an open or closed line such as length, angle, curvature, and others. SPC allows representing a given n-D point as a *single* 2D point losslessly by adapting shifts of pairs of coordinates as was shown above. CF and SF do not have such capability.

Next CF, SF, and GLC including PC are not invariant to the *order* of coordinates. Different orderings produce different figures in all of them. This is not necessarily a deficiency because humans can discover patterns in some visualizations easier than in others. Once a pattern is discovered visually in one of orderings of coordinates, it can be converted to the *analytical* form that is "order free." Formulas (1)–(4) in section "Advantages of GLC for real data visualization" provide an example. To see independence of these formulas from ordering of coordinates, it is sufficient just to substitute indexed labels X_i of the coordinate by their actual names such as age or income. Also, in GLC, coordinates can be labeled by actual names of attributes from the beginning. See Figure 10 for health monitoring. It avoids memory overload to remember the meaning of indexed labels of coordinates X_i. Both CF and SF require remembering meaning of visual features in terms of attributes they encode, because commonly they are not labeled.

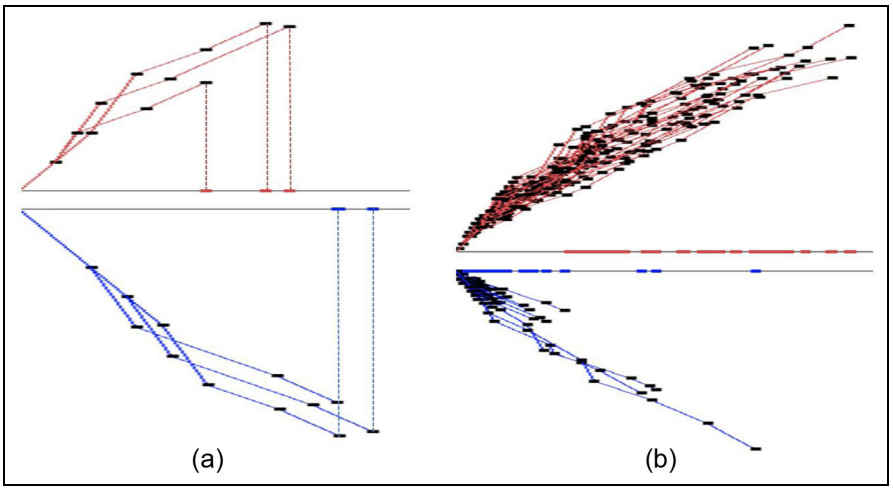

Figure 25. GLC-L algorithm on real data; (a) result with axis X_1 starting at axis U and repeated for the second class below it and (b) visualized data from two classes of Wisconsin breast cancer dataset from UCI ML Repository.

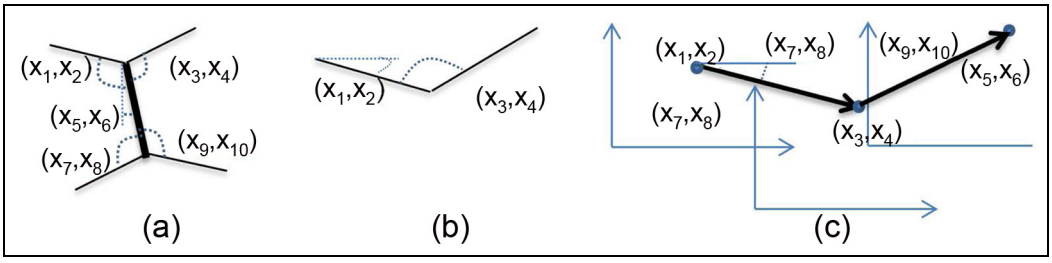

Figure 26. Stick figures and joint Shifted Paired Coordinates with stick figure; (a) stick figure with five sticks that encodes 10 attributes, (b) stick figure with two sticks that encodes four attributes, and (c) joint Shifted Paired Coordinates and stick figure with two sticks that encodes 10 attributes.

In CPC, graphs are directed, but in SF, the graphs are not directed. The directions of the graph edges can be beneficial, for example, it shows a trend in World hunger data in Kovalechuk.[12] One of the benefits of SF is familiarity of human body skeleton metaphor that can be remembered faster. On the other side, this metaphor limits the number of features that have a meaning in this metaphor, for example, arms, legs, and body.

Next, we propose a way to combine SF and paired GLC to *increase* the number of attributes to be encoded by the graph. It is based on the fact that SPC and SF use different parts of the graph to encode attributes (SPC use nodes and SF edges). The idea is to use *both nodes and edges for encoding attributes*. SPC do not use the length of the edges and angles between them to encode attributes, but use them for simplification of graphs as it is done in section "Methods for decreasing occlusion and pattern simplification." The length and the angles of the edges can be adjusted in SPC to make their values to represent attributes.

To get a desired length of the edge, a horizontal shifting of pairs of coordinates is sufficient. To get a desired angle of the edge shifting, a pairs of coordinates along a given radial distance from the node where edge is originated is sufficient. In this way, a graph with two arrows will encode not 6 attributes as in the SPC but 10 attributes as shown in Figure 26(c). SF to represent 10 attributes requires five edges (sticks; see Figure 26(a)) and with two edges it encodes only 4 attributes (see Figure 26(b)). While this method works for n-D points shown side-by-side as it is always done with SF, it does not work for drawing graphs of multiple n-D points in the same SPC space. The reason is that adjusting the length and the angles for the second n-D point changes the length and the angle for the first n-D point already adjusted.

Conclusion

This article shows that while preserving all multidimensional data in 2D visualization is a long-standing

problem, its solution is feasible beyond well-known Parallel and Radial (Star) Coordinates based on the concepts of *reversible lossless GLC*. Besides losslessness, the major advantage of GLC is multiplicity of GLC, which allows adjusting them to specific n-D data that are visualized. It is shown that GLC adjustment increases expressiveness of GLC including decreasing occlusion and simplifying visual pattern for the given n-D datasets. Health monitoring and Iris data classification tasks benefit from these capabilities. For Iris data, two-layer GLC visual methods exceeded accuracy of other visual methods (PC and Radvis). It also gets accuracy similar to the accuracy of SVM for Iris data, but with less overfitting risk than SVM. The results of the experimental part of the article with about 70 participants show the abilities of visual discovery of n-D patterns using GLC with n = 160 dimensions.

GLC capabilities for useful presentation of high-dimensional datasets

In many engineering application, 10% improvement in efficiency is considered as a valuable progress provided by a new technology. For GLC, the benchmarks of current technology are Parallel and Radial (Star) Coordinates. Relative to these methods, the progress in efficiency can be measured by decreasing the occlusion, which is indicated by decreasing the number of 2D points and lines per n-D point. As it is shown in this article and in prior publications,[12,14,16,49] such GLC as CPC, SPC, and Star CPC improve this measure two times (100%). Also, this article (section "Experimental comparison of visual recognition of 160D linear data structures with Gaussian noise in PC, Stars, and CPC Stars") and previous studies[14,49] show that GLC are capable representing data in 48D, 96D, 160D, and 192D where humans are capable discovering patterns for classification of these high-dimensional data. This shows the usefulness of GLC at the current stage of its development, including visual representation of Challenger Disaster and financial applications.[12,52]

The patterns representing the *relational information* in different GLC such as CPC, SPC, and Star CPC in comparison with PC are shown in section "Visual representation of n-D relations in GLC" and several other sections. Below we summarize these comparisons:

- PC is a *special case* of GLC when all coordinates are parallel. Thus, it is logical to use PC as one of GLC (not as opposing to GLC) where PC is *more intuitive and simpler* than other GLC in discovering relations. For instance, CPC can produce self-crossing (non-planar) graphs. PC, SPC, and Star CPC do not have this issue and can be applied when self-crossing in CPC hides the relation to be discovered.
- Each graph edge in CPC and SPC directly visualizes a relation of *four dimensions*. In PC, it directly visualizes only a relation between *two adjacent dimensions*.
- PC requires *four nodes* to represent a relation between four dimensions; CPC and SPC require *two nodes* with *twice less occlusion*.
- In PC, for each value x_i, a different line must be drawn to show the linear relations $x_j = mx_i + b$ for the two adjacent dimensions. For all values of x_i, this leads to an *infinite set of lines* for this linear relation. CPC and SPC allow a *single line* in a classical Cartesian form.
- In PC, the *infinite set of lines* for linear relations $x_j = mx_i + b$ (1D regression) creates an extreme case of *full occlusion* (no line visible). Therefore, this drawing is *not scalable* for large datasets. A *single line* in CPC and SPC for the same dataset has *no occlusion* and is *scalable*.
- Classical Cartesian visualization of linear relations $x_j = mx_i + b$ used CPC, and SPC is *familiar* to everyone. It *does not require learning a new visualization* in contrast with PC for this relation.
- Compact representations of the linear relation $y = kx + m$ (that do not directly map individual points x to y) have *the same expressiveness* in PC, CPC, and SPC requiring a single 2D point (section "2D linear dependencies in PC, CPC, and SPC").
- Figure 10 shows much *simpler* and *more familiar* SPC visualization of 4D health monitoring relations than in PC. It shows the relation between the initial health status and its change over time to the goal state.
- Figures 11 and 12 show *highly accurate* linear discrimination relations between Iris classes produced in SPC. PC and Radvis do not reveal such a linear discrimination relation (Figure 14).
- Figure 19 shows *more accurate* results with Star CPC (94%) than with PC (79%) in discovering noisy 160D linear relations by humans.
- Figure 23 shows *simpler, more familiar, and less occluded* visualization of a noisy 23D linear relation in CPC than in PC.
- Figure 25 shows GLC-L capabilities to represent *weighted discriminating linear relations* between n dimensions (section "Linear dependencies in GLC for classification and numerical estimates"). Such capabilities are not known for the PC.
- Commonly, non-linear relations are modeled by interpolating them by a set of linear relations. The

listed advantages of different GLC for linear relations with noise and showed an opportunity to use them for *interpolating non-linear relations* in the future.

The wider question for the further studies is how to make the progress further to higher dimensions and larger datasets. We envision three approaches.

The first approach that was briefly outlined in the introduction is a *hybrid approach*, which combines lossless and lossy methods. In this approach, first, a lossy method such as PCA is used to decrease, say 400 dimensions to 10 dimensions (first 10 principal components) with loss of information that is considered as acceptable. Then, GLC is used to visualize data in these 10 dimensions without any further loss of information. In contrast, the attempt to visualize 400D in the first two principal components directly without GLC will often lead to very significant loss of information.

The second approach is expanding the *GLC side-by-side approach* used in section "Experimental comparison of visual recognition of 160D linear data structures with Gaussian noise in PC, Stars, and CPC Stars," where each n-D point is shown as a separate graph (figure) preferably as a *closed contour* to leverage human perceptual abilities with closed contours. This visualization does not suffer from occlusion, but suffers from switching gazing from graph to graph. Also, while it dramatically increases the dimension to n = 160 and n = 192 as it is discussed in section "Experimental comparison of visual recognition of 160D linear data structures with Gaussian noise in PC, Stars, and CPC Stars," it limits the number of graphs analyzed at each given time by a few dozens of graphs.

The third approach is splitting dimensions by clustering them and visualizing data in each subset of dimensions separately with combining patterns found in such subsets of dimension to a joint pattern. For instance, data with 1000 dimension can be split into 10 clusters of 100 dimensions. All three approaches are topics of future exploration. While we expect progress in all of them, we do not expect that GLC will provide a "silver bullet" for all possible tasks and data as it is the case with all current methods.

In summary, this article (1) proposed new methods for decreasing occlusion and simplifying visual patterns for classification tasks, (2) demonstrated efficiency of new compact lossless representation by PSPC on real Iris and health monitoring data, (3) proposed a new two-layer GLC concept and demonstrated its efficiency on real data, (4) demonstrated advantages of closed contour lossless visual representations over PC for high-dimensional data in the experiment with several about 70 participants for classification of modeled data (linear hypertubes), and (5) clarified limits of high-dimensionality of data for human visual classification of modeled n-D data (linear hypertubes) in PC, Star CPC, and Radial Coordinates. This creates an opportunity to design advanced hybrid data mining/machine learning methods that integrate advantages of analytical and visual methods to get higher accuracy, interpretability, and avoiding overgeneralization and overfitting of discovered patterns. In the future, such hybrid exploration may provide end users with "n-D glasses" to conduct deep n-D data exploration with less extensive involvement of data scientists.

Acknowledgement

We are very grateful to the anonymous reviewers for their valuable feedback and helpful comments.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

References

1. Bertini E, Tatu A and Keim D. Quality metrics in high-dimensional data visualization: an overview and systematization. *IEEE T Vis Comput Gr* 2011; 17(12): 2203–2212.
2. Ward M, Grinstein G and Keim D. *Interactive data visualization: foundations, techniques, and applications*. Natick, MA: A K Peters, Ltd, 2010.
3. Inselberg A. *Parallel coordinates: visual multidimensional geometry and its applications*. New York: Springer, 2009.
4. Simoff S, Bohlen M and Mazeika A (eds). *Visual data mining*. Berlin; Heidelberg: Springer, 2008.
5. Tergan S and Keller T (eds). *Knowledge and information visualization*. Berlin; Heidelberg: Springer, 2005.
6. Keim DA, Hao MC, Dayal U, et al. Pixel bar charts: a visualization technique for very large multi-attribute data sets. *Inform Visual* 2002; 1(1): 20–34.
7. Wong P and Bergeron R. 30 years of multidimensional multivariate visualization. In: Nielson GM, Hagan H and Muller H (eds) *Scientific visualization—overviews, methodologies and techniques*. Washington, DC: IEEE Computer Society Press, 1997, pp. 3–33.
8. Hoffman P, Grinstein G, Marx K, et al. DNA visual and analytic data mining. In: *Proceedings of the visualization'97*, Phoenix, AZ, 18–24 October 1997, pp. 437–441. New York: IEEE.
9. Belianinov A, Vasudevan R, Strelcov E, et al. Big data and deep data in scanning and electron microscopies: deriving functionality from multidimensional data sets. *Adv Struct Chem Imag* 2015; 1(1): 6.
10. Kandogan E. Star coordinates: a multi-dimensional visualization technique with uniform treatment of dimensions. In: *Proceedings of the IEEE information visualization symposium*, Salt Lake City, UT, 9–10 October 2000, vol. 650, p. 22. New York: IEEE.

11. Kandogan E. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In: *Proceedings of the 7th ACM SIGKDD international conference on knowledge discovery and data mining*, San Francisco, CA, 26–29 August 2001, pp. 107–116. New York: ACM.
12. Kovalerchuk B. Visualization of multidimensional data with collocated paired coordinates and general line coordinates. In: *Proceedings of the IS&T/SPIE electronic imaging: visualization and data analysis*, San Francisco, CA, 2 February 2014, paper no. 90170I. Bellingham, WA: International Society for Optics and Photonics.
13. Grishin V and Kovalerchuk B. Multidimensional collaborative lossless visualization: experimental study. In: *Proceedings of the international conference on cooperative design, visualization and engineering*, Seattle, WA, 14–17 September 2014, pp. 27–35. Cham: Springer International Publishing.
14. Kovalerchuk B and Grishin V. Collaborative lossless visualization of n-D data by collocated paired coordinates. In: *Proceedings of the international conference on cooperative design, visualization and engineering* (Lecture notes in computer science 8683), Seattle, WA, 14–17 September 2014, pp. 19–26. Cham: Springer International Publishing.
15. Kovalerchuk B and Smigaj A. Computing with words beyond quantitative words: incongruity modeling. In: *Proceedings of the 2015 annual conference of the North American fuzzy information processing society (NAFIPS) held jointly with 2015 5th world conference on soft computing (WConSC)*, Redmond, WA, 17–19 August 2015, pp. 1–6. New York: IEEE.
16. Kovalerchuk B. Super-intelligence challenges and lossless visual representation of high-dimensional data. In: *Proceedings of the 2016 international joint conference on world computational intelligence congress, neural networks (IJCNN)*, Vancouver, BC, Canada, 24–29 July 2016, pp. 1803–1810. New York: IEEE.
17. Yang J, Peng W, Ward MO, et al. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. In: *Proceedings of the 9th annual IEEE conference on information visualization (INFOVIS 2003)*, Seattle, WA, 19–21 October 2003, pp. 105–112. New York: IEEE.
18. Johansson J and Forsell C. Evaluation of parallel coordinates: overview, categorization and guidelines for future research. *IEEE T Vis Comput Gr* 2016; 22(1): 579–588.
19. Jolliffe J. *Principal component analysis*. New York: Springer-Verlag, 1986.
20. Jäckle D, Fischer F, Schreck T, et al. Temporal MDS plots for analysis of multivariate data. *IEEE T Vis Comput Gr* 2016; 22(1): 141–150.
21. Kruskal J and Wish M. *Multidimensional scaling*. Thousand Oaks, CA: SAGE, 1978.
22. Mead A. Review of the development of multidimensional scaling methods. *J Roy Stat Soc D: Sta* 1992; 41: 27–39.
23. Sharko J, Grinstein G and Marx K. Vectorized Radviz and its application to multiple cluster datasets. *IEEE T Vis Comput Gr* 2008; 14(6): 1444–1427.
24. Daniels K, Grinstein G, Russell A, et al. Properties of normalized radial visualizations. *Inform Visual* 2012; 11(4): 273–300.
25. Kohonen T. *Self-organization and associative memory*. Berlin: Springer-Verlag, 1984.
26. Gorban AN, Kégl B, Wunsch DC, et al. (eds). *Principal manifolds for data visualization and dimension reduction*. Berlin: Springer, 2008.
27. Duch W, Adamczak R, Grąbczewski K, et al. *Extraction of knowledge from data using computational intelligence methods*. Toruń: Copernicus University, 2000, https://www.fizyka.umk.pl/~duch/ref/kdd-tut/Antoine/mds.htm
28. Rubio-Sánchez M, Raya L, Díaz F, et al. A comparative study between RadViz and Star coordinates. *IEEE T Vis Comput Gr* 2015; 22(1): 619–628.
29. Stahnke J, Dork M, Muller B, et al. Probing projections: interaction techniques for interpreting arrangements and errors of dimensionality reductions. *IEEE T Vis Comput Gr* 2015; 22(1): 629–638.
30. Lee JH, McDonnell KT, Zelenyuk A, et al. A structure-based distance metric for high-dimensional space exploration with multidimensional scaling. *IEEE T Vis Comput Gr* 2014; 20(3): 351–364.
31. Maestre E and Ramírez R. An approach to predicting bowing control parameter contours in violin performance. *Intell Data Anal* 2010; 14(5): 587–599.
32. Ma D, Huang J, Liu C, et al. Coupling model based on PCA-WNN for comprehensive evaluation of water quality. In: *Proceedings of the 2010 international conference on challenges in environmental science and computer engineering (CESCE)*, Wuhan, China, 6–7 March 2010, vol. 1, pp. 227–231. New York: IEEE.
33. Rosipal R, Girolami M, Trejo LJ, et al. Kernel PCA for feature extraction and de-noising in nonlinear regression. *Neural Comput Appl* 2001; 10(3): 231–243.
34. Grishin V. *Pictorial analysis of experimental data*. Moscow, Russia: Nauka Publishing, 1982, 237 pp. (in Russian).
35. Kovalerchuk B and Schwing J. *Visual and spatial analysis: advances in data mining, reasoning, and problem solving*. Dordrecht: Springer Science & Business Media, 2005.
36. Kovalerchuk B, Delizy F, Riggs L, et al. Visual data mining and discovery with binarized vectors. In: Holmes DE and Jain LC (eds) *Data mining: foundations and intelligent paradigms*. Berlin; Heidelberg: Springer, 2012, pp. 135–156.
37. Blascheck T, John M, Kurzhals K, et al. VA^2: a visual analytics approach for evaluating visual analytics applications. *IEEE T Vis Comput Gr* 2016; 22(1): 61–70.
38. Few S. Tapping the power of visual perception. *Vis Bus Intel Newsl* 2004; 39: 41–42.
39. Appelbaum LG and Norcia AM. Attentive and pre-attentive aspects of figural processing. *J Vision* 2009; 9(11): 18.
40. Theus M. High-dimensional data visualization. In: Chen C-H, Härdle WK and Unwin A (eds) *Handbook of data visualization*. Berlin; Heidelberg: Springer, 2008, pp. 151–178.
41. Dzemyda G, Kurasova O and Žilinskas J. *Multidimensional data visualization: methods and applications*. New York: Springer Science & Business Media, 2012.

42. Kaur P. Support Vector Machine (SVM) with Iris and mushroom dataset, 2016, http://image.slidesharecdn.com/svm-131205085208-phpapp01/95/support-vector-machinesvm-with-iris-and-mushroom-dataset-15-638.jpg?cb=1471176596
43. Mafrur R. SVM example with Iris Data in R, 2015, http://rischanlab.github.io/SVM.html
44. Trafimow D and Marks M. Editorial. *Basic Appl Soc Psychol* 2015; 37(1): 1–2.
45. Trafimow D and Rice S. A test of the null hypothesis significance testing procedure correlation argument. *J Gen Psychol* 2009; 136(3): 261–270.
46. Valentine JC, Aloe AM and Lau TS. Life after NHST: How to describe your data without "*p*-ing" everywhere. *Basic Appl Soc Psych* 2015; 37(5): 260–273.
47. Pickett R and Grinstein G. Iconographic displays for visualizing multidimensional data. In: *Proceedings of the 1988 IEEE international conference on systems, man, and cybernetics*, Beijing, China, 8–12 August 1988, pp. 514–519. New York: IEEE.
48. Grinstein G, Pickett R and Williams MG. EXVIS: an exploratory visualization environment. In: *Proceedings of the graphics interface '89*, London, ON, Canada, 19–23 June 1989, vol. 89, pp. 254–261. Canadian Information Processing Society.
49. Kovalerchuk B and Grishin V. Visual data mining in closed contour coordinates. In: *Proceedings of the IS&T international symposium on electronic imaging 2016, visualization and data analysis (VDA)*, San Francisco, CA, 14 February 2016, vol. VDA-503.1–VDA-503.10.
50. Fayyad UM, Wierse A and Grinstein GG. *Information visualization in data mining and knowledge discovery.* San Francisco, CA: Morgan Kaufmann, 2002.
51. Fua YH, Ward MO and Rundensteiner EA. Hierarchical parallel coordinates for exploration of large datasets. In: *Proceedings of the conference on visualization'99: celebrating ten years*, San Francisco, CA, 24–29 October 1999, pp. 43–50. New York: IEEE Computer Society Press.
52. Wilinski A and Kovalerchuk B. Visual knowledge discovery and machine learning for investment strategy. *Cogn Syst Res* 2017; 44: 100–114.
53. Vityaev E and Kovalerchuk B. Visual data mining with simultaneous rescaling. In: Kovalerchuk B and Schwing J (eds) *Visual and spatial analysis: advances in data mining, reasoning, and problem solving*. Dordrecht: Springer Science & Business Media, 2005, pp. 371–386.
54. Rubio-Sánchez M , Raya L, Díaz F, et al. A comparative study between RadViz and Star Coordinates. *IEEE T Vis Comput Gr* 2015; 22: 619–628. DOI: 10.1109/TVCG.2015.2467324.

Appendix 1

For convenience of reading this article, the appendix below summarizes results from Kovalerchuk[16] and Appelbaum and Norcia[39] that are used in this article. While GLC are constructed by drawing n coordinate axes in 2D in a variety of ways, curved, parallel, unparalleled, collocated, disconnected, and so on, this construction must be accompanied by algorithms and mathematical statements for constructing a 2D graph that are described in sections "Algorithms for graphs in general line coordinates" and "Statements" based on Kovalerchuk[16] where the proof of the statements can be found. Section "Preattentive aspects of straight horizontal and vertical line processing" summarizes the experiment on preattentive perception of straight lines from Appelbaum and Norcia.[39]

Algorithms for graphs in general line coordinates

Algorithm 1. Constructing a graph as a collection of directed edges (arrows, vectors). Each edge is located on the respective coordinate X_i starting at the origin of this coordinate and ending at point x_i on X_i (see Figure 27). It is called a *basic GLC graph constructing algorithm* (GLC-B).

Algorithm 2. Constructing a graph by connecting location of x_i on X_i with the location of x_{i+1} on X_{i+1}, starting from $i = 1$ and ending at $i = n$ (see Figure 27(b)). This is a generalization to GLC of the algorithm used in Parallel Coordinates (PC).[3] It is called *GLC-PC graph constructing algorithm*.

Algorithm 3. Constructing a graph by the algorithm as illustrated in Figure 27(c). It moves the start point of each vectors $\mathbf{x}_{i+1}$ to the end of vector $\mathbf{x}_i$. This algorithm is a generalization to GLC of the algorithm implemented in the special Star Coordinates (SC).[10,11] It is called *GLC-SC graph constructing algorithm*.

Algorithm 4. Constructing a graph by the algorithm that is illustrated in Figure 27(d). It is called *GLC-CC graph constructing algorithm*, where 2D points produced in consecutive pairs (X_i, X_{i+1}) are connected by directed edges. It is a generalization to GLC of the algorithm implemented in the Collocated Coordinates (CC) shown in Figure 1(d)–(f) in section "Background definitions."

Figure 27 shows that algorithm 4 requires three points and two lines, but algorithm 1 requires 12 points and 6 lines for lossless representation of an n-D point. Algorithm 2 requires six points and five lines, and algorithm 3 requires seven points and six lines. In general, algorithm 4 (GLC-CC) requires two times less points and lines than algorithms 1–3. This is *fundamental advantage* of GLC-CC algorithm from a human cognitive viewpoint, because it simplifies pattern discovery by a naked eye. Below algorithms 1 and

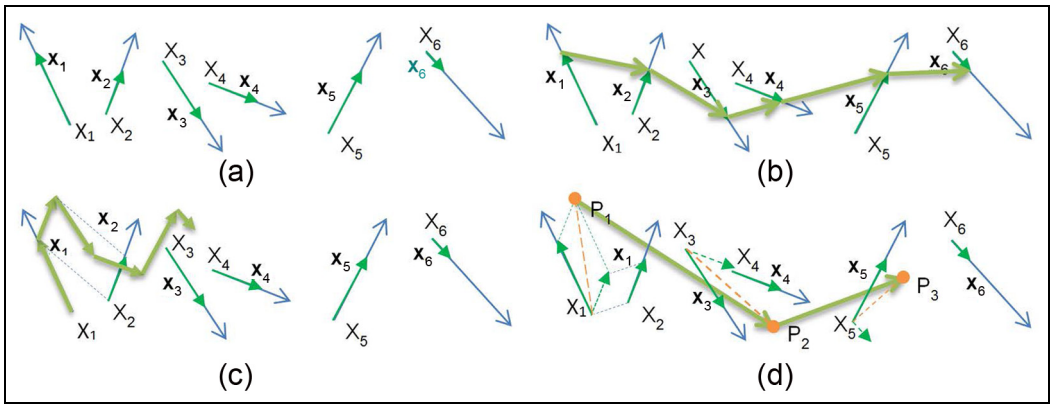

Figure 27. 6D data point (0.75, 0.5, 0.7, 0.6, 0.7, 0.3) in different coordinate systems; (a) six coordinates and six vectors that represent a 6D data point (0.75, 0.5, 0.7, 0.6, 0.7, 0.3), (b) 6D data point (0.75, 0.5, 0.7, 0.6, 0.7, 0.3) in GLC-PC, (c) 6D data point (0.75, 0.5, 0.7, 0.6, 0.7, 0.3) in GLC-SC, and (d) 6D data point (0.75, 0.5, 0.7, 0.6, 0.7, 0.3) in GLC-CC.

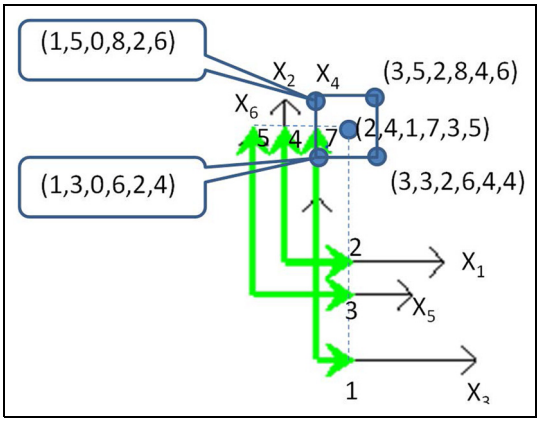

Figure 28. Data in parameterized Shifted Paired Coordinates. Blue dots are corners of the square S that contains all graphs **N** of all n-D points of hypercube H for 6D base point (2, 4, 1, 7, 3, 5) with distance 1 from this base point.

4 are presented more formally as a set of steps for graph generation.

Basic GLC graph construction algorithm (GLC-B):

Step 1. Build GLC (see Figure 27(a) for an example with n = 6).
Step 2. Select an n-D point, for example, (7, 5, 6, 5, 6, 2).
Step 3. For each i (i = 1:n) locate value x_i in the coordinate X_i (see Figure 27(a) for example), and define n vectors $\mathbf{x}_i$ of length x_i from the origin of X_i that we denote as O_i.

GLC-CC graph construction algorithm:

Step 1. Construct vectors $\{\mathbf{x}_i\}$ by using basic GLC-B algorithm.

Step 2. Compute the sum of vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, $\mathbf{x}_{12} = \mathbf{x}_1 + \mathbf{x}_2$ and then compute the point $P_1 = O_1 + \mathbf{x}_{12}$.

Next, sum vectors $\mathbf{x}_3$ and $\mathbf{x}_4$, $\mathbf{x}_{34} = \mathbf{x}_1 + \mathbf{x}_2$, and get the point $P_2 = P_1 + \mathbf{x}_{34}$. Repeat this process for all next i. For even n, the last point is $P_{n/2} = P_{n/2-1} + \mathbf{x}_{n-1,\,n}$ (see Figure 27(d)), for odd n, the last point is $P_{(n+1)/2} = P_{(n+1/2)-1} + 2\mathbf{x}_n$.

Step 3. Build a directed graph by connecting points: $P_1 \rightarrow P_2 \cdots \rightarrow P_n$. This graph can be closed with $P_n \rightarrow P_1$.

Statements

Statement 1. The graph constructed by the GLC-CC algorithm has one-to-one mapping to n-D point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and has less than a half of the nodes and edges than GLC-PC and GLC-SC.

Figure 28 illustrates this property. It shows a cognitive advantage of the GLC-CC representation, which has a twice smaller footprint in 2D, relative to GLC-PC and GLC-SC. This results in a much *smaller occlusion* when multiple n-D data are visualized in 2D. Another advantage of it is the ability to represent *losslessly* any n-D point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ as *a single 2D point instead of a graph*. The algorithm to produce this representation is called the *Single Point (SP) algorithm*.

Steps of the *Single Point (SP) algorithm*:

Step 1. Select an arbitrary 2D point $A = (a_1, a_2)$ on the plane in some 2D Cartesian coordinate system (U_1, U_2) with origin (O_{u1}, O_{u2}). Point A is called the *2D anchor point* in (U_1, U_2). Then, select the n-D point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ that will be called the *base n-*

D point (n-D anchor). Next, select a set of positive constants $c_1, c_2, ..., c_n$ that will be used as lengths of coordinates $X_1, X_2, ..., X_n$.
Step 2. Compute 2D points $O_1 = (a_1 - x_1, a_2 - x_2)$ and $E_1 = (a_1 - x_1 + c_1, a_2 - x_2)$ in (U_1, U_2). Coordinate line X_1 is defined as vector (O_1, E_1) in (U_1, U_2).
Step 3. Define points $O_2 = O_1$ and $E_2 = (a_1 - x_1, a_2 - x_2 + c_2)$ in (U_1, U_2). Coordinate line X_2 is defined as vector (O_2, E_2) in (U_1, U_2).
Step 4. Repeat steps 2 and 3 for all the next pairs $(X_3, X_4), ..., (X_{n-1}, X_n)$ to build the coordinate system $X_1, X_2, ..., X_n$.

This algorithm creates the *Parametric Shifted Paired Coordinates* (PSPC) system, where each next pair of coordinates is drawn in the shifted Cartesian coordinates. These coordinates are defined by *parameters* which are respective components of the n-D anchor/base point **x** and the 2D anchor point A.

Statement 2. In the coordinate system $X_1, X_2, ..., X_n$ constructed by the Single Point algorithm with a given n-D anchor/base $\mathbf{x} = (x_1, x_2, ..., x_n)$ and anchor 2D point A, the n-D point **x** is mapped one-to-one to a single 2D point A by GLC-CC algorithm. The point (2, 4, 1, 7, 3, 5) in Figure 28 is an example of this statement.

Another advantage of the combination of GLC-CC and Single Point algorithms is that all n-D points of an n-D hypercube around a given n-D anchor/base point $\mathbf{x} = (x_1, x_2, ..., x_n)$ are mapped to graphs that are located within a square defined by the square algorithm that is defined below.

Steps of Square algorithm:

Step 1. Construct a hypercube H with center at the base point $\mathbf{x} = (x_1, x_2, ..., x_n)$ and distance d to its faces. Respectively, 2^n nodes **N** of this hypercube are $(x_1 + \alpha d, x_2 + \alpha d, ..., x_n + \alpha d)$, where $\alpha = 1$ or $\alpha = -1$ depending on the node, for example, $(x_1 + d, x_2 + d, ..., x_n + d)$, $(x_1 - d, x_2 - d, ..., x_n - d)$, $(x_1 + d, x_2 - d, ..., x_n - d)$.
Step 2. Construct a square S around point (a_1, a_2) with corners: $(a_1 + d, a_2 + d)$, $(a_1 + d, a_2 - d)$, $(a_1 - d, a_2 + d)$, $(a_1 - d, a_2 - d)$.

Statement 3 (locality statement). All graphs **N** that represent nodes of hypercube H are within square S. Figure 28 illustrates this statement and its proof.

Preattentive aspects of straight horizontal and vertical line processing

Below we provide a summary of the experimental electroencephalogram (EEG) study in Appelbaum and Norcia[39] on preattentive perception. The goal one of the experiments in this work was measuring time required for people to distinguish vertical and horizontal lines in the central region from the lines on the background (see Figure 29). The lines in the center were changed with frequency that differs from the frequency of change of the background. This difference allowed to measure time for processing straight lines in the center. Specifically, the lines in the center were updated at 3 Hz and that of the background at 3.6 Hz. Each straight line is represented as a one-dimensional random luminance bar with a minimum bar width of 6 arc min. Participants were asked to detect subtle changes in a set of straight lines (Figure 29(b)). The participants indicated a change with a button press. On 20% of the 1.67-s stimulus cycles, the aspect ratio (horizontal to vertical) became elliptical versus circular at the beginning as Figure 29(b) shows. Responses were monitored and the aspect ratio was adjusted to

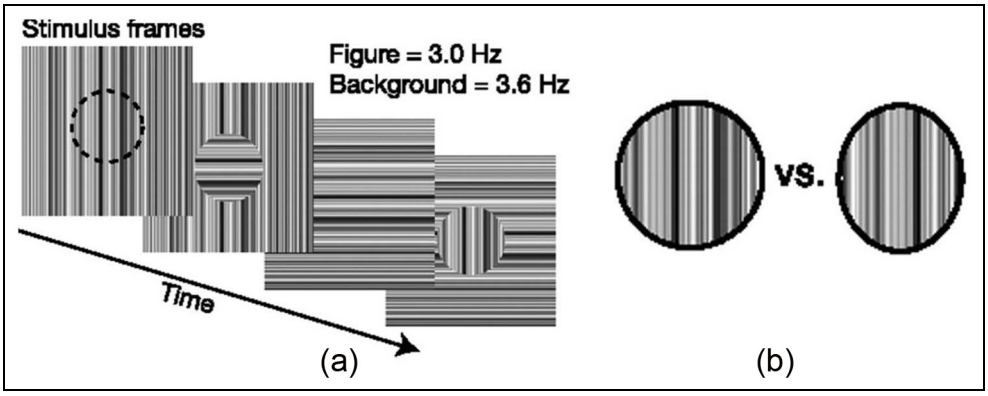

Figure 29. Preattentive aspects of straight horizontal and vertical line processing;[39] (a) difference in frequency of change of stimulus and background and (b) shape discrimination.

maintain performance at approximately 80% correct detection.

These authors assessed the amplitude and timing of brain responses involved in sets of straight lines in the center versus background processing. They concluded that collectively, their results indicate that basic aspects of scene segmentation of separating straight lines from the background proceed *preattentively*. This conclusion is based on the result of statistical analysis of the electroencephalogram collected with a whole-head, 128-channel, Geodesic EEG system with HydroCell Sensor Nets. Note that this result covers many straight lines. In our examples, in Figures 5–8, just a few straight lines are used, that is, a much simpler case that respectively also should be preattentive and can be generalized for more lines.