Available online at www.sciencedirect.com

ScienceDirect

Cognitive Systems Research 45 (2017) 95–108

Cognitive Systems RESEARCH

www.elsevier.com/locate/cogsys

# Visual cognitive algorithms for high-dimensional data and super-intelligence challenges

Boris Kovalerchuk

*Dept. of Computer Science, Central Washington University, USA*

Received 4 February 2017; accepted 28 May 2017
Available online 6 June 2017

**Abstract**

In the long run the cognitive algorithms intend to make super-intelligent machines and super-intelligent humans. This paper presents a technical process to reach specific aspects of super-intelligence that are out of the current human cognitive abilities. These aspects are inabilities to discover patterns in large numeric multidimensional data with a naked eye. This is a long-standing problem in Data Science and Modeling in general. The major obstacle is in human inability to see n-D data by a naked eye and our needs in visualization means to represent n-D data in 2-D losslessly. While these means exist their number and abilities are limited. This paper expands the class of such lossless visual methods, by further developing a new concept of Generalized Shifted Paired Coordinates. It shows the advantages of proposed reversible lossless technique by representing real data and by proving mathematical properties.


*Keywords:* Cognitive algorithms; High-dimensional data; Visualization; Machine learning, generalized coordinates, super-intelligence

## 1. Introduction

The concept of *human-machine super-intelligence* is present in the literature for a long time (Hibbard, 2002). It includes prospects of both *super-intelligent machines* and *super-intelligent humans* that will far surpass the current human intelligence significantly lifting the human cognitive limitations.

The expected ways to achieve it range from progress in: (1) Artificial Intelligence (AI) and Computational Intelligence (CI), (2) new human abilities to evolve or directly modify their biology (Superintelligence), and (3) power of crowd interaction (Michelucci & Dickinson, 2015). A significant portion of publications in this area is the futuristic predictions of when super-intelligence can be achieved, and what the potential danger of expected achievements is. This paper is devoted to the different aspect, namely, a technical way to reach some specific aspects of super-intelligence that are beyond the current human cognitive abilities. It is to overcome inabilities to analyze a large amount of abstract numeric *high-dimensional* data and finding complex *patterns* in these data with a *naked eye*.

Human inability to discover patterns in n-D data using a naked eye is one of the major motivations for the emergence of visual analytics research area that is devoted to developing 2-D visual representations (visualizations) of n-D data. While multiple such representations have been developed, many of them are lossy, i.e., do not represent n-D data completely and do not allow restoring n-D data completely from their 2-D representation. Respectively our abilities to discover n-D data patterns from such incomplete 2-D representations are limited and potentially erroneous.

In contrast lossless visualizations of n-D data have no such limitations and have advantages as *cognitive enhancers*

*E-mail address:* borisk@cwu.edu

http://dx.doi.org/10.1016/j.cogsys.2017.05.007
1389-0417/

of the human cognitive abilities to discover n-D data patterns. Below we review the state of the art in this area, and outline the challenges that this paper addresses. Discovering patterns in big multidimensional data using visual means is a long-standing problem in Information Visualization, Visual Analytics, Visual Data Mining, and Data Science in general (Bertini, Tatu, & Keim, 2011; Grishin & Kovalerchuk, 2014; Hoffman & Grinstein, 2002; Inselberg, 2009; Kovalerchuk, 2014; Kovalerchuk & Grishin, 2014; Simov, Bohlen, & Mazeika, 2008; Tergan & Keller, 2005; Ward, Grinstein, & Keim, 2010; Wong & Bergeron, 1997). As we already outlined the major challenge is our *cognitive limitations*. We cannot see n-D data by a naked eye and need enhanced visualization tools ("n-D glasses") to represent n-D data in 2-D losslessly.

The number of available tools to overcome this cognitive limitation is quite limited. Principal Component Analysis (PCA) is a lossy n-D data representation when we use the first two main principal components to show n-D data in 2-D. Multidimensional scaling is also a lossy representation due to approximation of n-D distances. Simple tools such as heat maps, pie-and bar-graphs are applicable to relatively small datasets and dimensions. Parallel Coordinates (PC) and Radial (star) Coordinates (RC) today are the most known lossless n-D data visualization methods for relatively large data while suffering from occlusion.

There is a need to *extend* the class of lossless n-D data visual representations. A new class of such representations called the **General Line Coordinates (GLC)** and several of their specifications have been proposed in Grishin and Kovalerchuk (2014), Kovalerchuk (2014), and Kovalerchuk and Grishin (2014). These visualizations include Collocated Paired Coordinates (CPC) in orthogonal and radial forms. The benefits of these new visual representations and their advantages have been shown in Grishin and Kovalerchuk (2014), Kovalerchuk (2014), and Kovalerchuk and Grishin (2014) for analyzing data of the Challenger disaster, World Hunger, Semantic shift in humorous texts, and others.

This paper: (1) expands these new methods, (2) explores their *mathematical properties*, and (3) demonstrates advantages of these methods for *real-world data*. In exploration of the mathematical properties, we analyze how the methods represent known n-D data structures in 2-D. The importance to explore the mathematical properties of new methods in addition to comparing them with known methods on real-world data is in the ability to derive general properties that are common to all data of a given structure.

This paper is organized as follows. Section 2 presents the concept of lossless visualization of n-D data as cognitive enhancer for discovering n-D data patterns. Section 3 provides definitions of line coordinates. Section 4 provides algorithms and mathematical statements that demonstrate how n-D data representations in various general line coordinates simplify representation of n-D data in 2-D for better perceptual and cognitive abilities for visual pattern discovery. Section 5 shows advantages of different GLCs on real-world data. Section 6 demonstrates the technique to simplify the visual patterns in GLC. Section 7 relates super-intelligence issues to high-dimensional data. The paper is concluded with its summary and discussion of future studies.

## 2. Lossless visualization of n-D data as cognitive enhancer for discovering patterns

This paper described the proposed new algorithms for lossless visual representation of high-dimensional data and their connections with human super-intelligence challenges. We interpret these algorithms as cognitive algorithms that enhance human cognitive abilities to deal with modern Big data high-dimensional challenges. The paper focused on Generalized Shifted Paired Coordinates as a subset of General Line Coordinates. The advantages of these coordinates have been shown both mathematically and on the data. These advantages guide future studies to solve a major challenge. This challenge is finding conditions for a provable property of simpler and less overlapped lossless 2-D representation of the non-intersecting hyper-ellipses, hyper-rectangles, and other shapes in n-D.

The advantage of a wide class of General Line Coordinates is that it allows multiple different visualizations of the same data with the different perceptual and cognitive characteristics. This multiplicity increases the chances that humans will be able to reveal the hidden n-D patterns in these visualizations. It is not realistic to expect that a single visualization will do this for all possible data and all humans.

A full classification of the general line coordinates for the cognitively efficient n-D data visualization is a task for future research as well as the deeper links with Machine Learning to be able to build visually the learning algorithms using visual means in GLC such as Decision Trees. This is an area of future studies for the design of more complete processes and for expanding to other data mining/-machine learning methods.

Example. Assume that we have established that new data have the same mathematical structure that was explored before. Then we can use the derived matched structural properties. Consider n-D data with a mathematical structure where all n-D points of class $C_1$ are in the one hypercube and all n-D points of class $C_2$ are in another hypercube and the distance between these hypercubes is greater or equal to k lengths of these hypercubes.

Assume that it was established mathematically that for any n-D data with this structure a lossless visualization method $V_1$, produces visualizations of n-D vectors of classes $C_1$ and $C_2$ that do *not overlap* in 2-D. Next assume that this property was tested on new n-D data and was confirmed. In this case we can apply visualization method $V_1$ with confidence that it will produce desirable visualization without occlusion of two classes. Similarly if the structural property is negative to ability to visualize the pattern with-

Table 1
Line coordinates.

| Type | Characteristics |
|---|---|
| General Line Coordinates (GLC) | Drawing n coordinate axes in 2-D in a variety of ways: curved, parallel, unparalleled, collocated, disconnected, etc. |
| Collocated Paired Coordinates (CPC) in 2-D | For each n-D point **x** splitting it into pairs of its coordinates $(x_1, x_2), \ldots, (x_{n-1}, x_n)$; drawing each pair as 2-D point in the same two axes on the plane and linking these 2-D points to form an directed graph |
| Collocated Paired Coordinates in 3-D | Splitting n coordinates into triples and representing each triple as 3-D point in the same three axes; and linking these points to form an directed graph for each n-D point |
| Basic 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, i.e., coordinates shifted to the location of the first pair of a given n-D point |
| Partially Collocated Coordinates | Drawing some coordinate axes in 2D collocated and some coordinates not co-located |
| 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 an n-gon one after another |

out occlusion then this will lead to the conclusion that the method should not be used for the given data.

## 3. Definitions of line coordinates

This section provides necessary background information on different forms of General Line Coordinates mostly from Kovalerchuk (2014). GLC contains the well-known parallel and radial (star) coordinates and the new ones listed in Table 1, which generalize them by locating coordinates in any place, direction, and in any topology (connected or disjoined). Figs. 1–3 show the examples of General Line Coordinates.

**In-Line Coordinates** (ILC) shown in Fig. 2d are similar to parallel coordinates, except that the axes $X_1$, $X_2$, … $X_n$ are horizontal, not vertical. Each pair is represented as a Bezier Curve. The height of the curve is the distance between the two adjacent values, e.g., for (5,4,0,6,4,10), the heights are 1, 4, 6, 2, 6.

The algorithm for representing the n-D points in 2-D using lossless **collocated paired coordinates (CPC)** (see Fig. 2a) is presented below. We use an example in 6-D with a state vector x = (x, y, x′, y′, x″, y″), here x and y are the location of the object, x′ and y′ are velocities (derivatives), and x″ and y″ are accelerations (second derivatives) of this object.

The main steps of the algorithm are:

- Normalization of all dimensions to some interval, e.g., [0,1];
- Grouping attributes into consecutive pairs (x, y) (x′, y′) (x″, y″);
- Plotting each pair in the same orthogonal normalized Cartesian coordinates X and Y, and
- Plotting a directed graph (x, y) → (x′, y′) → (x″, y″) with directed paths from (x, y) to (x′, y′) and from (x′, y′) to (x″, y″).

Fig. 2a shows the application of this algorithm to a 6-D 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).

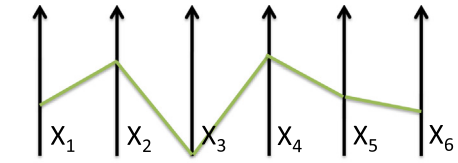
The basic **Shifted Paired Coordinates** (SPC) show each next pair in the shifted coordinate system. The first pair (5,4) is drawn in the (X, Y) system. The next pair (0,6) is drawn not in the original system (X, Y), but in the shifted coordinate system denoted as (X + 1, Y + 1), where coordinate X is shifted up by 1, and coordinate Y is shifted to the right by 1. This means 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 the 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 Fig. 2b. The **generalized SPC** allow shifting each coordinate to any value. These shifts are parameters of SPC and when the parameters are given SPC are called **parameterized SPC** (PSPC).

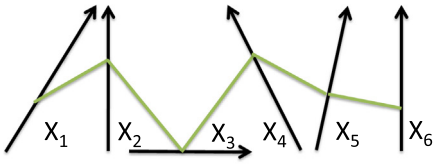
The **Anchored Paired Coordinates** (APC) represent each next pair starting at the first pair that serves as an "anchor" with plotting an arrow from point (x, y) to point (x + x′, x + y′) and from point (x + x′, x + y′) to point (x + x″, x + y″).

The graph of a 6-D point (1,1,1,1,1,1) in **Partially Collocated Radial Coordinates** is shown in Fig. 3 on the left as a blue[1] triangle. The same 6-D point in the **Cartesian Collocated Paired Coordinates** on the right produced a much simpler graph as a single point. Fig. 3 illustrates the perceptual and cognitive differences between the alternative 2-D representations of the same n-D data. Here a 2-D point

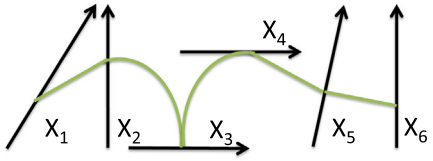
[1] For interpretation of color in Figs. 3, 9, 11, 12, 16, and 17, the reader is referred to the web version of this article.



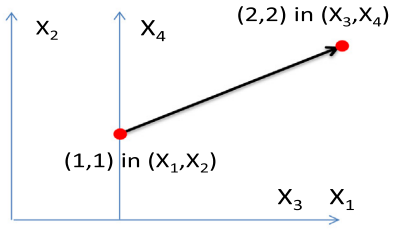

(a) 6-D point in Parallel Coordinates



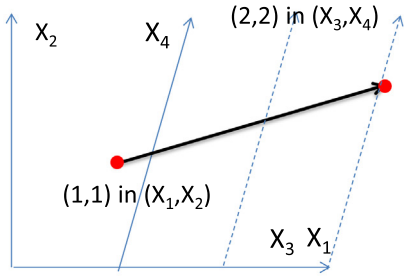

(b) 6-D point in General Line Coordinates with straight lines



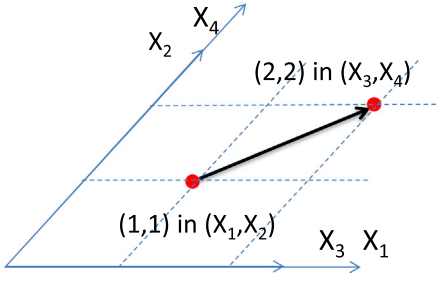

(c) 6-D point in General Line Coordinates with curves




(d) Partially Collocated Orthogonal (Ortho) Coordinates



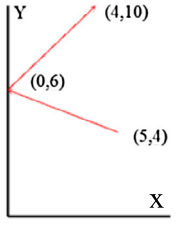

(e) Partially Collocated Ortho non-Ortho Coordinates



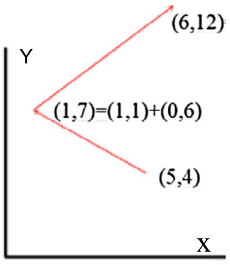

(d) Collocated non-Ortho Coordinates

Fig. 1. Examples of General Line Coordinates. (d-f) 4-D point (1,1,2,2) in different coordinate systems.

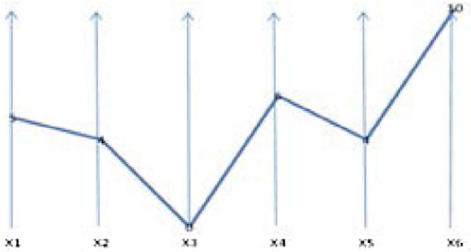
is much simpler perceptually and cognitively than a triangle for the same 6-D point.



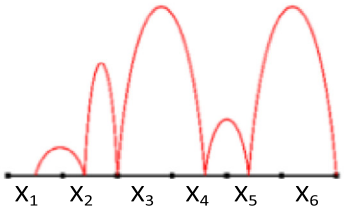

(a) 4-D point in collocated paired coordinates




(b) 4-D point in Shifted Paired Coordinates Point (6,12) is produced by adding (2,2) to (4,10).




(c) the same 4-D point as above in Parallel Coordinates




(d) the same 4-D point as above in In-Line Coordinates

Fig. 2. Data point (5,4,0,6,4,10) in different coordinate systems.

## 4. Graphs in general line coordinates

Next we give a more formal description of General Line Coordinates in vector algebra terms. While GLC axes can be drawn in 2-D in a variety of ways shown above it must be accompanied by an algorithm for constructing a 2-D graph that will represent an n-D point in these coordinates. We designed several such algorithms. Next, we name them and then specify their steps.

**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 Fig. 4a. We will call this algorithm a *basic GLC graph constructing algorithm* (GLC-B).

**Algorithm 2.** Constructing a graph by connecting the 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 Fig. 4b. This is a generalization to GLC of the algorithm implemented in Parallel Coordinates (PC) (Inselberg, 2009). Respectively we will call it as *GLC-PC graph constructing algorithm*.

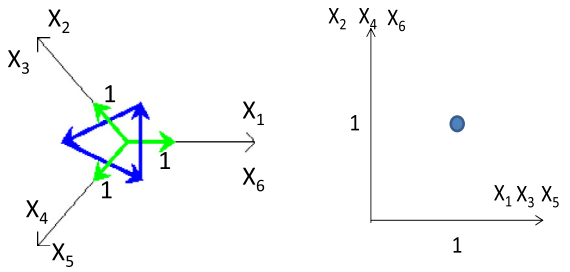



Fig. 3. 6-D point (1, 1, 1, 1, 1, 1) in two $X_1$-$X_6$ coordinate systems (left – in Radial Collocated Coordinates, right – in Cartesian Collocated Coordinates).

**Algorithm 3.** Constructing a graph by the algorithm as illustrated in Fig. 4c. It moves the start point of each of the 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 Star Coordinates (SC) (Kandogan, 2001). Respectively we will call it as *GLC-SC*1 *graph constructing algorithm*.

**Algorithm 4.** Constructing a graph by the algorithm which is illustrated in Fig. 4d. It is a generalization to GLC of the algorithm implemented in the Collocated Coordinates (CC) (Kovalerchuk, 2014) shown in Fig. 1d-f. We will call is this algorithm the *GLC-CC*1 *graph constructing algorithm*.

**Algorithm 5.** Constructing a graph by the algorithm that is illustrated in Fig. 4e. It is another generalization to GLC of the algorithm implemented in the Collocated Coordinates (CC) (Kovalerchuk, 2014) shown in Fig. 1d-f in combination with the idea of Algorithm 3, i.e., moving the next vector to the end of the previous one. We will call is this algorithm the *GLC-CC*2 *graph constructing algorithm*.

**Algorithm 6.** Constructing a graph by the algorithm as illustrated in Fig. 4f. It moves the *end* of the vector $\mathbf{x}_2$ to the *end* of vector $\mathbf{x}_1$ and for each other vectors $\mathbf{x}_{i+1}$ it moves its *start* point to the *end* point of vector $\mathbf{x}_i$, e.g., the *start* point of $\mathbf{x}_3$ is moved to the *end* point of vector $\mathbf{x}_2$. This algorithm is a generalization to GLC of the algorithm implemented in the Star Coordinates (SC) (Kandogan, 2001) and is a modification of Algorithm 3. Respectively we will call it as *GLC-SC*2 *graph constructing algorithm*.

Fig. 4 shows that Algorithms 4 and 5 require 3 points and 2 lines, but Algorithm 1 requires 12 points and 6 lines for lossless representation of an n-D point. Algorithm 2 requires 6 points and 5 lines, and Algorithm 3 requires 7 points and 6 lines. In general, Algorithm 4 (GLC-CC1) requires two times less points and lines than algorithms 1–3. This is a *fundamental advantage* of GLC-CC algorithm from human cognitive viewpoint, because it simplifies pattern discovery by a naked eye. Below we present Algorithms 1 and 4 more formally as a set of steps for graph generation.

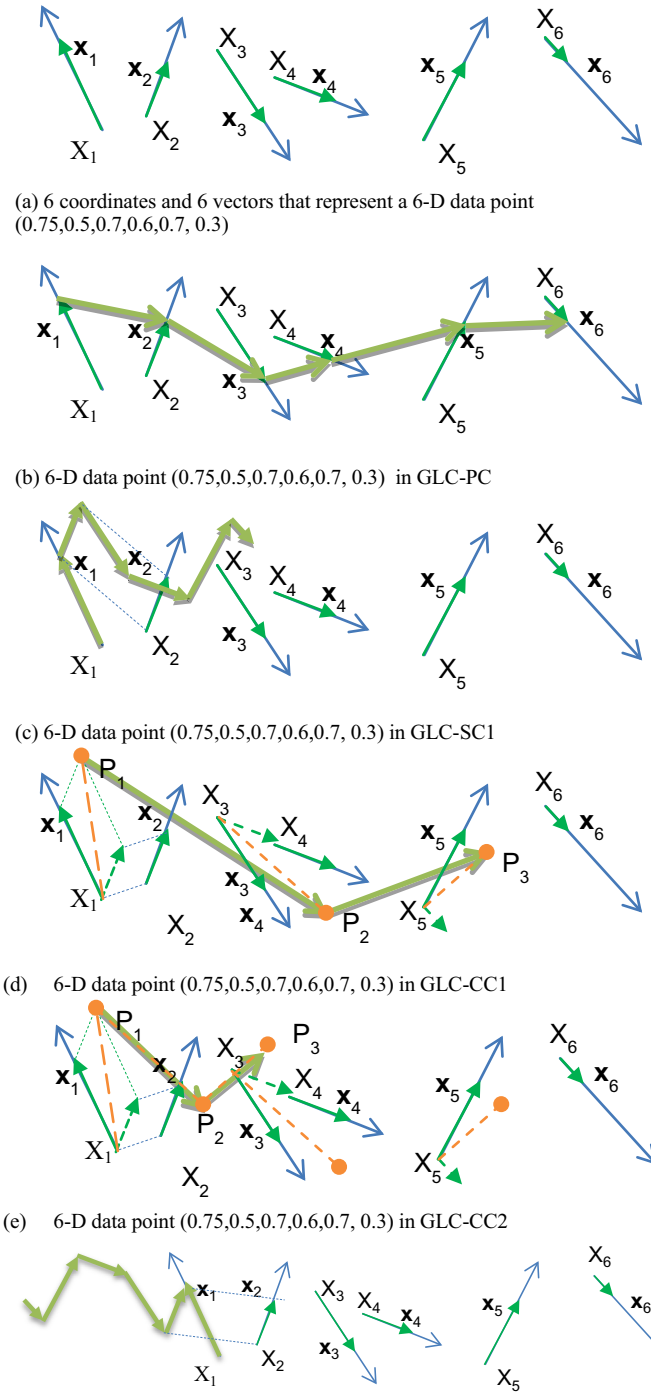



Fig. 4. 6-D data point (0.75, 0.5, 0.7, 0.6, 0.7, 0.3) in different coordinate systems.

**Basic GLC graph construction algorithm** (GLC-B)

Step 1: Build GLC (see Fig. 5 for an example with n = 6).
Step 2: Select an n-D point, e.g., (7, 5, 6, 5, 6, 2).
Step 3: For each i (i = 1:n) locate value $x_i$ in the coordinate $X_i$ (see Fig. 4a for an 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-CC1 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 compute the sum of vectors $\mathbf{x}_3$ and $\mathbf{x}_4$, $\mathbf{x}_{34} = \mathbf{x}_1 + \mathbf{x}_2$ and the point $P_2 = O_3 + \mathbf{x}_{34}$. Repeat this process by computing $P_3 = O_5 + \mathbf{x}_{56}$ and for all next *i*,
$P_i = O_{2i-1} + \mathbf{x}_{2i-1,2i}$. For even *n* the last point is $P_{n/2} = O_{n-1} + \mathbf{x}_{n-1,n}$ (see Fig. 4d), for odd *n* the last $\mathbf{x}_{n-1,n} = \mathbf{x}_n$ and the last point $P_{(n+1)/2} = O_n + \mathbf{x}_n$.
Step 3: Build a directed graph by connecting points $\{P\}$: $P_1 \to P_2 \to \ldots P_{i-1} \to P_i \ldots \to P_n$. This graph can be closed by adding edge $P_n \to P_1$.

**GLC-CC2 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 compute the sum of vectors $\mathbf{x}_3$ and $\mathbf{x}_4$, $\mathbf{x}_{34} = \mathbf{x}_1 + \mathbf{x}_2$ and the point $P_2 = P_1 + \mathbf{x}_{34}$. Repeat this process by computing $P_3 = P_2 + \mathbf{x}_{56}$ and for all next *i*, $P_i = P_{i-1} + \mathbf{x}_{2i-1,2i}$. For even *n* the last point is $P_{n/2} = P_{n/2-1} + \mathbf{x}_{n-1,n}$ (See Fig. 4d), for odd *n* $\mathbf{x}_{n-1,n} = \mathbf{x}_n$ and the last point is $P_{(n+1)/2} = P_{(n+1)/2-1} + \mathbf{x}_n$.
Step 3: Build an directed graph by connecting points $\{P\}$: $P_1 \to P_2 \to \ldots P_{i-1} \to P_i \ldots \to \ldots P_n$. This graph can be closed by adding edge $P_n \to P_1$.

**Statement.** The graph constructed by the GLC-CC1 algorithm has one-to-one mapping to n-D point $(x_1, x_2, \ldots x_n)$ and has no more than a half of the nodes and edges of GLC-PC and GLC-SC.

**Proof.** The point $P_1$ allows us to restore $x_1$ value by projecting $P_1$ to coordinate $X_1$ as shown in Fig. 4d. Formally it can be computed by representing the coordinate $X_1$ as a vector $\mathbf{X}_1$, and using a dot product $(P_1\text{-}O_1) \bullet \mathbf{X}_1$ of $\mathbf{X}_1$ with vector $(P_1\text{-}O_1)$. This gives us value $x_1$ and vector $\mathbf{x}_1$. Next, the property $P_1 = O_1 + \mathbf{x}_{12} = O_1 + \mathbf{x}_1 + \mathbf{x}_2$ allows us to compute $\mathbf{x}_2 = P_1\text{-}O_1 - \mathbf{x}_1$. The same can be done by a dot product $(P_1\text{-}O_1) \bullet \mathbf{X}_2$. In the same way by projecting point $P_2$ to $X_3$ we get $\mathbf{x}_3$ and then using $P_2 = O_3 + \mathbf{x}_{34} = O_3 + \mathbf{x}_3 + \mathbf{x}_4$ we restore $\mathbf{x}_4 = P_2\text{-}O_1 - \mathbf{x}_3$. These steps are continued for all points $P_i$ until all $x_i$ are restored.

The property of less than a half of the nodes and edges in GLC-CC1, relative to GLC-PC and GLC-SC, follows directly from their definitions. Fig. 4d illustrates this property. □

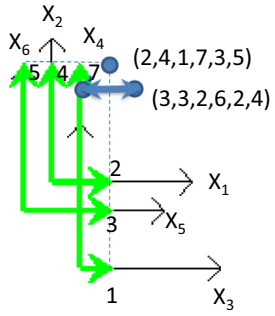



Fig. 5. 6-D points (3,3,2,6,2,4) and (2,4,1,7,3,5) in $X_1$-$X_6$ coordinate system build using point (2,4,1,7,3,5) as an anchor.

**Statement.** The graph constructed by the GLC-CC2 algorithm has one-to-one mapping to n-D point $(x_1, x_2, \ldots x_n)$ and has no more than a half of the nodes and edges of GLC-PC and GLC-SC.

**Proof.** The restoration of $\mathbf{x}_1$ and $\mathbf{x}_2$ is the same as in GLC-CC1. To get $\mathbf{x}_3$ and $\mathbf{x}_4$, we use the property $P_2\text{-}P_1 = \mathbf{x}_{34} = \mathbf{x}_3 + \mathbf{x}_4$. Next we project $\mathbf{x}_{34}$ to $\mathbf{X}_3$ and to $\mathbf{X}_4$ and $\mathbf{x}_3$ and $\mathbf{x}_4$. These steps are continued for all the points $P_i$ until all $\mathbf{x}_i$ are restored. The property of no more than a half of the nodes and edges in GLC-CC2, relative to GLC-PC and GLC-SC, also follows directly from their definitions. Fig. 4 illustrates this property too.

So far we had shown a cognitive advantage of both GLC-CC representations, i.e., its **twice smaller footprint in 2-D**, relative to GLC-PC and GLC-SC. This leads to **much smaller occlusion** when multiple n-D data are represented in 2-D. Below we show its other **advantage** – the ability to represent losslessly any n-D point $(x_1, x_2, \ldots, x_n)$ as a **single 2–D point instead of a graph**. The algorithm to produce this representation will be called the Single Point (SP).

Steps of the **Single Point algorithm**.

Step 1: Select an arbitrary 2-D point $A = (a_1, a_2)$ on the plane. This point will be called the **anchor 2-D point**. Then select the n-D point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ that will be called the **base n-D point** (or n-D anchor point). Next select a set of positive constants $c_1, c_2, \ldots, c_n$ that will be used a lengths of coordinates $X_1, X_2, \ldots, X_n$.
Step 2: Compute 2-D points $O_1 = (a_1\text{-}x_1,\ a_2\text{-}x_2)$ and $E_1 = (a_1\text{-}x_1\text{+}c_1,\ a_2\text{-}x_2)$. Coordinate line $X_1$ is defined as vector $\mathbf{X}_1 = (O_1, E_1)$.
Step 3: Define the points $O_2 = O_1$ and $E_2 = (a_1\text{-}x_1,\ a_2\text{-}x_2\text{+}c_2)$. Coordinate line $X_2$ is defined as a vector $\mathbf{X}_2 = (O_2, E_2)$.
Step 4: Repeat the steps 2 and 3 for all other coordinates to build the coordinate system $X_1, X_2, \ldots, X_n$.

This algorithm creates a **Parametrized Shifted Paired Coordinate** (**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 a base n-D point **x** and 2-D anchor point A. See Fig. 5. □

**Statement.** In the coordinate system $X_1, X_2, \ldots, X_n$ constructed by the Single Point algorithm with the given base n-D point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and the anchor 2-D point A, the n-D point **x** is mapped one-to-one to a single 2-D point A by GLC-CC algorithm.

**Proof.** Consider coordinate $X_1$ and a point located on $X_1$ at the distance $x_1$ from $O_1$. According to Step 2 of SP algorithm $O_1 = (a_1\text{-}x_1,\ a_2\text{-}x_2)$. Thus it is the point $(a_1\text{-}x_1+x_1,\ a_2\text{-}x_2) = (a_1,\ a_2\text{-}x_2)$. It is projection of pair $(x_1, x_2)$ to $X_1$ coordinate.

Similarly consider coordinate $X_2$ and a point located on $X_2$ at the distance $x_2$ from $O_1$. According to Step 2 of SP algorithm $O_2 = (a_1\text{-}x_1,\ a_2\text{-}x_2)$. Thus it is the point $(a_1\text{-}x_1,\ a_2\text{-}x_2+x_2) = (a_1\text{-}x_1,\ a_2)$. It is projection of pair $(x_1, x_2)$ to $X_2$ coordinate. Therefore, pair $(x_1, x_2)$ is represented in $X_1, X_2$ coordinate system as $(a_1, a_2)$. In the same way the pair $(x_3, x_4)$ is also mapped to the point $(a_1, a_2)$. The repeat of this reasoning for all next pairs $(x_i, x_{i+1})$ will match them to the same point $(a_1, a_2)$ too. This concludes the proof. See Fig. 5 that illustrates this proof for a 6-D point $(2,4,1,7,3,5)$.

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

In other words informally, n-D locality is converted to 2-D **locality** and wise versa, or, an n-D point **y** is close to the base n-D point **x** if and only if the graph of **y** is close to 2-D anchor point A.

**Steps of the Square algorithm**.

Step 1: Construct a hyper-cube H with the center at the base point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and distance d to its faces. Respectively $2^n$ nodes **N** of this hypercube are $(x_1+\alpha d, x_2+\alpha d, \ldots, x_n+\alpha d)$, where $\alpha = 1$ or $\alpha = -1$ depending on the node, e.g., $(x_1+d, x_2+d, \ldots, x_n+d)$, $(x_1\text{-}d, x_2\text{-}d, \ldots, x_n\text{-}d)$, $(x_1+d, x_2\text{-}d, \ldots, x_n\text{-}d)$.

Step 2: Construct a square S around point $(a_1, a_2)$ with corners: $(a_1+d, a_2+d)$, $(a_1+d, a_2\text{-}d)$, $(a_1\text{-}d, a_2+d)$, $(a_1\text{-}d, a_2\text{-}d)$. □

**Statement (locality statement).** All graphs **N** that represent nodes of hypercube H are within square S.

**Proof.** Consider the n-D node $(x_1+d, x_2+d, \ldots, x_n+d)$ of H where d is added to all coordinates of the n-D point X. This node is mapped to the 2-D point $(a_1+d, a_2+d)$ which is a corner of the square S. Similarly the node $(x_1\text{-}d, x_2\text{-}d, \ldots, x_n\text{-}d)$ of H where d is subtracted from all coordinates of x is mapped to the 2-D point $(a_1\text{-}d, a_2\text{-}d)$ which is another corner of the square S. In the same way the n-D node of the hypercube that contains pairs $(x_1+d, x_2\text{-}d)$, $(x_3+d, x_4\text{-}d), \ldots, (x_i+d, x_{i+1}\text{-}d), \ldots, (x_{n-1}+d, x_n\text{-}d)$, i.e., with positive d for the odd coordinates $(X_1, X_3, \ldots)$ and negative d for the even coordinates $(X_2, X_4, \ldots)$ is mapped to the 2-D point $(a_1+d, a_2\text{-}d)$. Similarly, a node with alternation of positive and negative d in all such pairs $(x_i\text{-}d, x_{i+1}+d)$ will be mapped to $(a_1\text{-}d, a_2+d)$. Both these points are also corners of the square S.

If an n-D node of H includes two pairs such as $(x_i+d, x_{i+1}+d)$ and $(x_j+d, x_{j+1}\text{-}d)$ then it is mapped to the graph that contains two 2-D nodes $(a_1+d, a_2+d)$ and $(a_1+d, a_2\text{-}d)$ that are corners of the square S. Similarly if an n-D node of H includes two other pairs $(x_k\text{-}d, x_{k+1}+d)$ and $(x_m\text{-}d, x_{m+1}\text{-}d)$ it is mapped to the graph that contains two 2-D nodes $(a_1\text{-}d, a_2+d)$ and $(a_1\text{-}d, a_2\text{-}d)$ that are two other corners of the square S. At most a hypercube's n-D node has all these four types of pairs that can be present several times in it, and respectively all of them will be mapped to four corners of the 2-D square S.

These corners are the 2-D nodes of the graph that represents this n-D node of the hypercube. Respectively, all edges of this graph will be within the square S.

Any other n-D point **y** of the hypercube H has at least one coordinate that is less than this coordinate for some node Q of this hypercube. For example, let $y_1 < q_1$ and $y_i = q_i$ for all other i, then all pairs $(y_i, y_{i+1})$, but the first pair $(y_1, y_2)$ will be mapped to the corners of the square S. The first pair $(y_1, y_2)$ will be mapped to the 2-D point, which is inside of the square S because $y_1 < q_1$. This concludes the proof. Figs. 6 and 7 illustrate this statement and its proof.

Both the Collocated Paired Coordinates and the Parametrized Shifted Paired Coordinates are **lossless,** and represent a **similar n-D point as similar 2-D graphs**, i.e., 2-D nodes of similar n-D points are located closely as Figs. 8 and 9 illustrate.

Fig. 8 shows an example of 4-D data of two classes in Collocated Paired Coordinates in blue and green ellipses. Fig. 9 shows data from Fig. 8 in the Parametrized Shifted

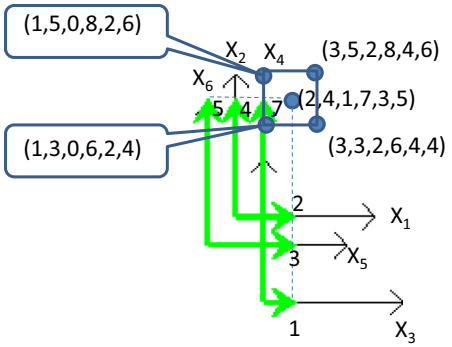



Fig. 6. Data in Parametrized Shifted Paired Coordinates. Blue dots are corners of the square S that contains all graphs **N** of all n-D points of hypercube H for 6-D base point $(2,4,1,7,3,5)$ with distance 1 from this base point. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

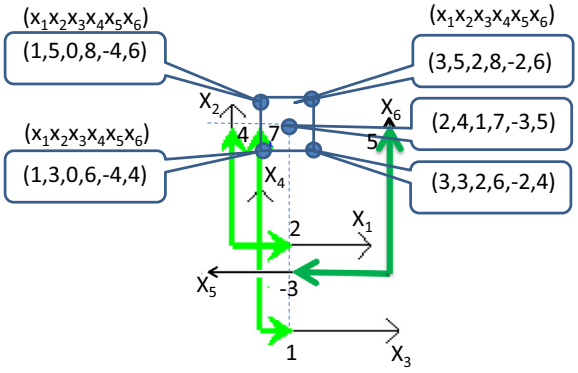



Fig. 7. Data in Parametrized Shifted Paired Coordinates. Blue dots are corners of the square S that contains all graphs **N** of all n-D points of hypercube H for 6-D base point $(2,4,1,7,-3,5)$ with distance 1 from this base point. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

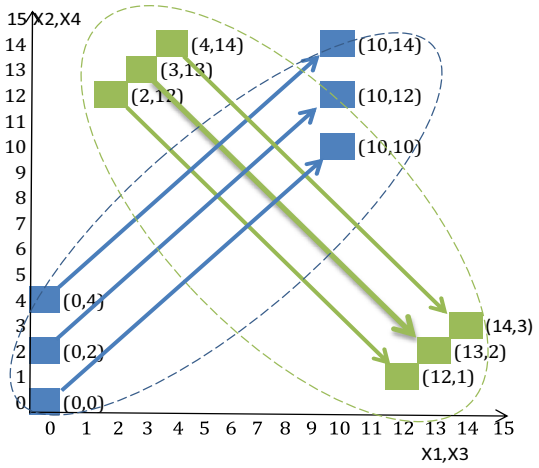



Fig. 8. 4-D data of two classes in Collocated Paired Coordinates shown in blue and green ellipses. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Paired Coordinates with 4-D point $(3,13,13,2)$ from the green class as the base point for parametrized shift.

Both Figs. 8 and 9 show the separation of two classes, but in Fig. 9, the separation between these blue and green classes is much simpler than in Fig. 8. This is a demonstration of the promising **advantages** of parametrized shifted coordinates to **simplify visual patterns** of n-D data in 2-D in tasks such as clustering and supervised classification.

This gives the direction for future studies to solve a major challenge. This challenge is finding the conditions where this empirical observation can be converted into the provable property of simpler and less overlapped 2-D representation of non-intersecting hyper-ellipses, hyper-rectangles, and other shapes in n-D. □

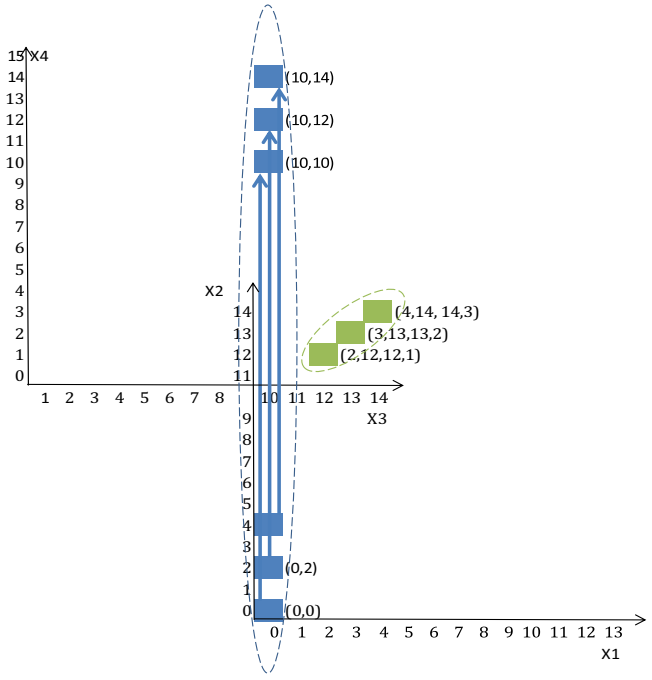



Fig. 9. 4-D data of two classes in Parametrized Shifted Paired Coordinates.

## 5. Real data visual analysis

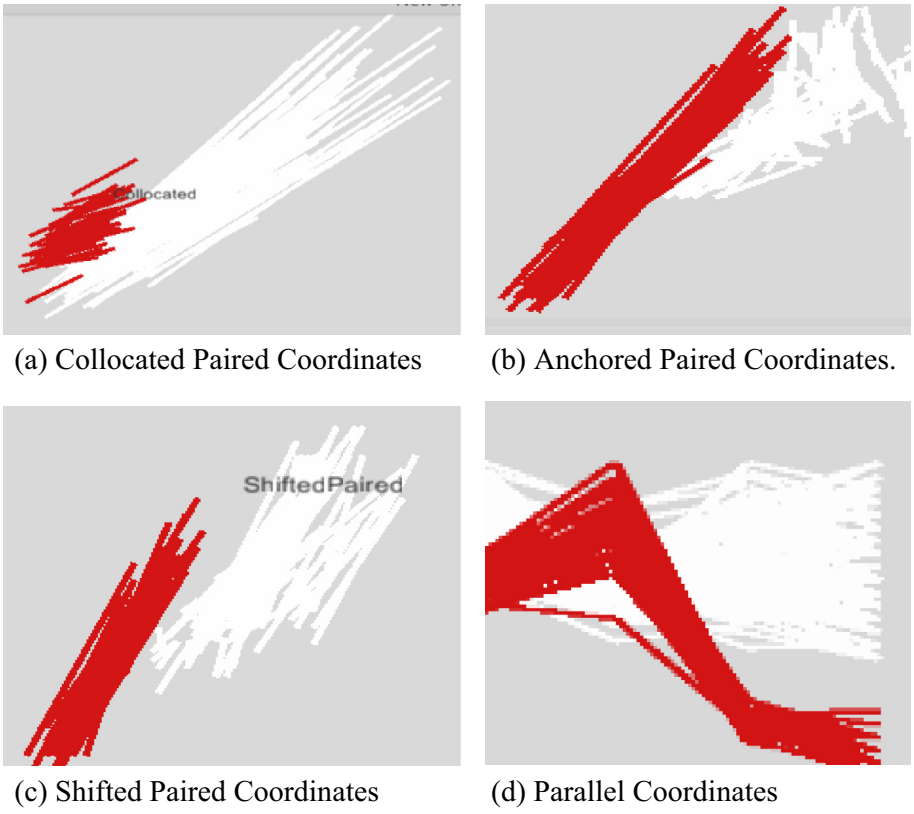
### 5.1. Iris data classification

Fig. 10 shows the results of the comparison of all four coordinates for the Iris data (Machine Learning Repository, 2017) that contain 150 4-D iris records. In contract with the Parallel Coordinates (Fig. 10d), the new Collocated Paired visualizations (Fig. 10a-c) practically have no overlap for these data. The iris-setosa class is clearly separated from the other two classes in these new visualizations. Note that these visualizations need only one 2-D segment to represent a 4-D data record. In contrast the Parallel Coordinates require three segments per 4-D record. The larger number of segments leads to more overlaps among the lines in parallel coordinates. Other successful experiments with real world data are presented in Kovalerchuk (2014) and Kovalerchuk and Grishin (2014).

### 5.2. Visual n-D feature extraction

While Fig. 10 shows the situations, where the visualization in General Line Coordinates systems (a)-(c) allow separate classes directly there are situations where more stages are needed. Fig. 11 shows 748 cases of 4-D data points of two classes from the Blood Transfusion Service Center (Machine Learning Repository, 2017) visualized in PSPC with the 4-D anchor point in the average point of the red class. Each 4-D point is represented losslessly as an arrow. In this PSPC space two classes and their convex hulls heavily overlap.

While this PSPC representation does not separate classes it gives multiple *visual insights* for extraction features that can be more discriminative for these classes. An observer can discover the differences in orientation




(a) Collocated Paired Coordinates (b) Anchored Paired Coordinates.

(c) Shifted Paired Coordinates (d) Parallel Coordinates

Fig. 10. Iris data (red Iris-setosa class). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

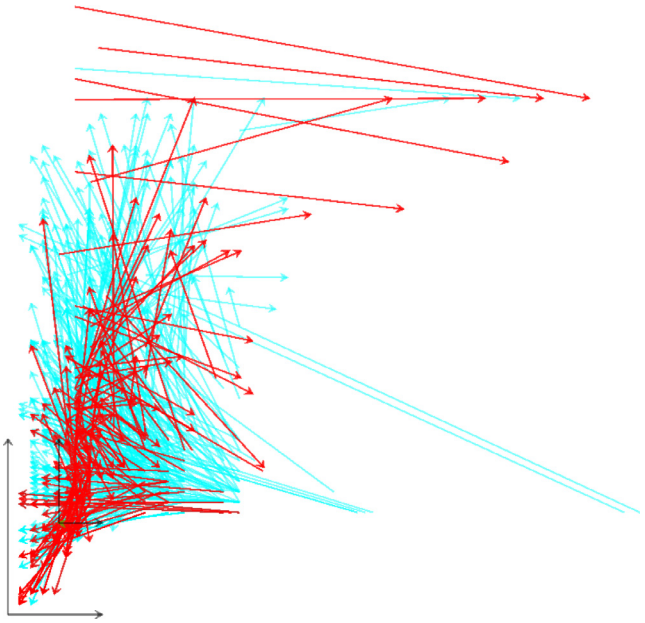


Fig. 11. Original 4-D data of two classes in PSPC system.

and length of some arrows of two classes. For instance, west and southwest orientation is common for a large number of red arrows in the bottom that are often shorter that blue arrows with the same direction. The length of the majority of the red arrows is also smaller than the length of the blue arrows with a few exclusions on the top. Therefore the next step is extracting two new features from PSPC: orientation, $y_1$, and length, $y_2$. The formulas for these new features are given below in terms of two variables,

$$v_1 = x_3 + (a_1 - a_3) - x_1, v_2 = x_4 + (a_2 - a_4) - x_2,$$

where $x_1$, $x_2$, $x_3$ and $x_4$ are values of coordinates of a given arrow **x** in Fig. 11, and ($a_1$, $a_2$, $a_3$, $a_4$) are coordinates of the 4-D anchor point A which is the average point of the red class.

In these terms a new features $y_1$ and $y_2$ are defined as follows:

$y_1$: if $v_2 \geq 0$ then $y_1 = \arccos(v_1/v_3)$ else $y_1 = 2\pi - y_1$
$y_2$: $y_2 = (v_1^2 + v_2^2)^{1/2}$.

Then both $y_1$ and $y_2$ are normalized to $[0, 1]$ interval for the visualization. The result is shown in Fig. 12 as a standard scatter plot of these two attributes. It shows that the majority of the blue cases are in the middle and the upper section of this middle part contains mostly the blue cases. The most overlapping area is the section of the lower section of the middle part. This informal visual exploration gives an insight on the area, where the next stage of feature extraction must be concentrated. This is the area with most of the overlap in Fig. 12. The cases from that area can be visualized in the original PSPC system to attempt to extract

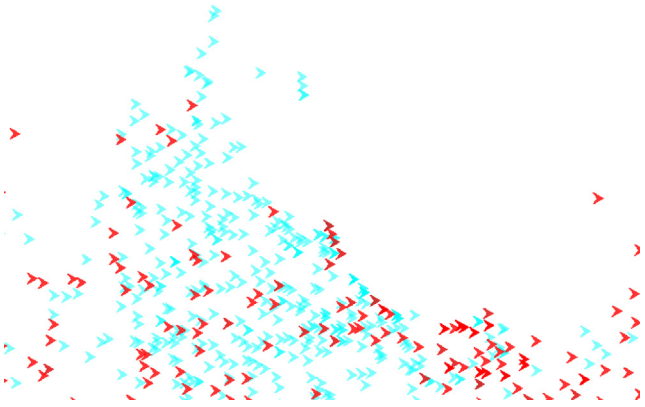


Fig. 12. Two classes in features $y_1$ and $y_2$ extracted using visual insight.

additional features that can separate blue and read cases in this smaller dataset.

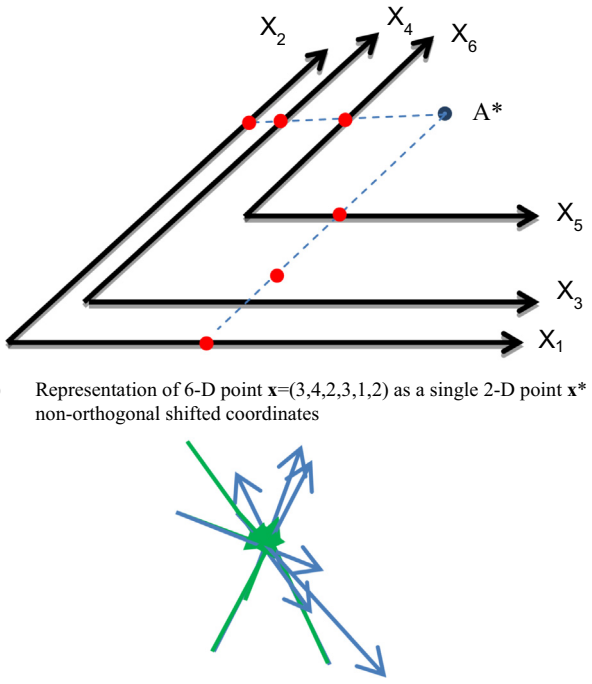
What are the chances to extract features $y_1$ and $y_2$ pure analytically using analytical machine learning techniques without any visual insight? From our viewpoint it is not realistic. It would require: (1) to identify a class of functions {f} that will include both functions $y_1$ and $y_2$ without knowing that these functions can be potentially useful, (2) to develop a formal criterion K that will evaluate all functions from {f} to be a good feature, (3) to have significant computing resources to run K on {f}. To ensue (1) we would need to make the class of functions {f} very large if not infinite one. This makes (2) and especially (3) extremely unrealistic. This example illustrates the general advantages of visualization approach that in large part substitutes cognition by perception (Munzner, 2015).

## 6. Generalization of a fixed point to GLC

Other general line coordinates also allow the creating of a single 2-D point from an n-D point A losslessly. Fig. 13a illustrates it for the *non-orthogonal shifted paired coordinates.*

**Statement.** Any n-D general line coordinates (including the connected or disconnected, the orthogonal or non-orthogonal) constructed by Algorithms 1 and 2 can be shifted to represent a given n-D point as a single 2-D point losslessly.

**Proof.** Consider a 2-D point A, coordinate axes $X_1$-$X_n$ and values $(x_1, x_2, \ldots, x_n)$ on the an n-D point **x** that drawn in some 2-D coordinate system $(U_1, U_2)$ in accordance with Algorithm 1. Let $u(O_{i1}, O_{i2})$, $u(E_{i1}, E_{i2})$ be respectively coordinates of the origin and the end of axis $X_i$ in $(U_1, U_2)$ coordinate system, and pair $(u_{i1}, u_{i2})$ be a location in $(U_1, U_2)$ where vector $\mathbf{x}_i$ ends on the axis $X_i$, i.e., represents the value $x_i$. Coordinates of point A are given as $(a_1, a_2)$ in coordinates $(U_1, U_2)$. Consider a vector $\mathbf{v}_i = (a_1\text{-}u_{i1}, a_2\text{-}u_{i2})$ that represents the difference between points A and $(u_{i1}, u_{i2})$ that represents value $x_i$. Shifting axis $X_i$ by this




(a) Representation of 6-D point **x**=(3,4,2,3,1,2) as a single 2-D point **x*** in non-orthogonal shifted coordinates



(b) Six coordinates $X_1$-$X_6$ with 6 vectors on them that represent a 6-D data point (0.75,0.5,0.7,0.6,0.7, 0.3) shifted to a single point $A=(a_1,a_2)$.

Fig. 13. Representation of 6-D points as single 2-D point losslessly.

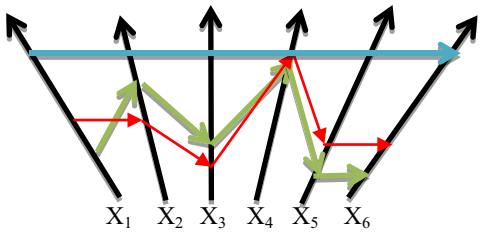
difference will bring the end point of vector $\mathbf{x}_i$ from $(u_{i1}, u_{i2})$ to A: $(a_1\text{-}u_{i1}, a_2\text{-}u_{i2}) + (u_{i1}, u_{i2}) = (a_1, a_2)$. Doing this for all axes $X_i$ will move all $x_i$ to A. The result of this shifting process for $X_1$-$X_6$ from Fig. 4a is shown in Fig. 13b.

While the given n-D point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ will be encoded as a single 2-D point, other n-D points visualized according to Algorithm 2 (illustrated in Figs. 4b and 14a) will still have n nodes and n-1 edges in the graph as in Fig. 4b. However the graphs of the n-D points that are close to the given n-D point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ will be smaller (see Fig. 14b).

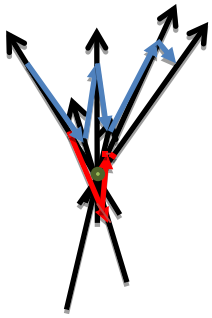
When this shifting is applied to the coordinates with collinear coordinates (see Fig. 15a) we use curvy lines to connect the points to be able to see n-D points as shown in Fig. 15b for blue and read 6-D points. □

**Statement.** Any n-D general line coordinates (including the connected or disconnected, orthogonal or non-orthogonal) constructed by Algorithms 1 and 4 can be shifted to represent a given n-D point as a single 2-D point losslessly.

**Proof.** Below we use the same notation and approach as in the proof for Algorithms 1 and 2. The difference is that instead of shifting end points of vectors $x_i$ we shift points $P_i = (p_{i1}, p_{i2})$ to $A = (a_1, a_2)$. This is done by shifting a whole pair of coordinates that forms $P_i$. by vector $\mathbf{w}_i = (a_1\text{-}p_{i1}, a_2\text{-}p_{i2})$. This will bring point $P_i$ to A: $(p_{i1}, p_{i2}) + (a_1\text{-}p_{i1}, a_2\text{-}p_{i2}) = (a_1, a_2)$.# □

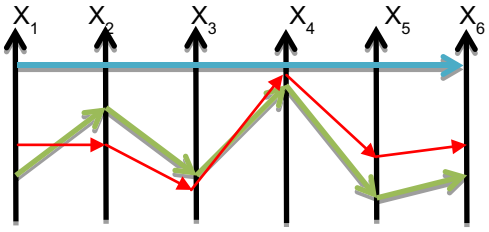



(a) Three 5-D points in GLC that generalize parallel coordinates

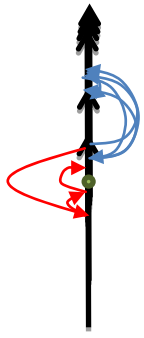


(b) GLC with coordinates $X_1$,$X_2$,$X_4$,$X_5$,$X_6$ shifted to $X_3$. The green line from (a) is collapsed to a single 2-D point (small green circle).

Fig. 14. Three 5-D points in GLC before and after shifting. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)




(a) Three 5-D points in parallel coordinates



(b) Shifted parallel coordinates $X_1$,$X_2$,$X_4$,$X_5$,$X_6$ to $X_3$. The green line from (a) is collapsed to a single 2-D point (small green circle).

Fig. 15. Three 5-D points in parallel GLC before and after shifting. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Statement.** Any n-D general line coordinates (including connected or disconnected, orthogonal or non-orthogonal) constructed by Algorithms 1 with any of Algorithms 3, 5 and 6 *cannot* be shifted to represent a given n-D point as a single 2-D point losslessly.

**Proof.** In algorithms 3, 5, and 6 in contrast with Algorithms 2 and 4 the length of each edge of the graph $\mathbf{x}^*$ of the n-D point $\mathbf{x}$ is fixed and cannot be changed under shifting. This prevents the collapsing any two nodes of $\mathbf{x}^*$ to a single node that is needed for getting a single 2-D point representing $\mathbf{x}$. Below we use the same notation and approach as in the proof for Algorithms 1 and 2.

The difference from the previous proof is that instead of shifting end points of vectors $x_i$ we shift points $P_i = (p_{i1}, p_{i2})$ to $A = (a_1, a_2)$. This is done by shifting a whole pair of coordinates that forms $P_i$. by vector $\mathbf{w}_i = (a_1\text{-}p_{i1}, a_2\text{-}p_{i2})$. This will bring point $P_i$ to A:$(p_{i1}, p_{i2}) + (a_1\text{-}p_{i1}, a_2\text{-}p_{i2}) = (a_1, a_2)$. □

## 7. Simplification process: examples

Cognitive load can be significantly decreased when a more complex visualization of the same data is simplified. Figs. 16–23 illustrate this process for different GLCs. All these figures show the same four 6-D points: A = (0.3, 0.6, 0.4, 0.8, 0.2, 0.9) (thick blue line) and C = (0.8, 0.9, 0.4, 0.5, 0.2, 0.3) (thick green line) and two other 6-D points (thin lines) of blue and green classes. Points A and C can be viewed as representative points (centers) of these classes. We start from representing these data in Parallel Coordinates in Fig. 16. This representation is quite complex with 6 points and 5 zigzag lines for every 6-D point without clear visual separation between points of two classes. In contrast representation in Shifted Paired Coordinates (SPC) in Fig. 17 is simpler with 3 points and two lines for each 6-D point. However the lines of two classes cross each other in ($X_1$, $X_2$), ($X_3$, $X_4$) spaces.

A simpler pattern would be if lines do not cross in these spaces. This is implemented in Fig. 18 by substituting coordinate $X_4$ to 1-$X_4$. While it removes one it creates another crossing in ($X_3$, 1-$X_4$) and ($X_5$, $X_6$) spaces.

This crossing is resolved in Fig. 19 by substituting the coordinate $X_6$ by 1-$X_6$. Now the cases of two classes are separated, but still not preattentive.

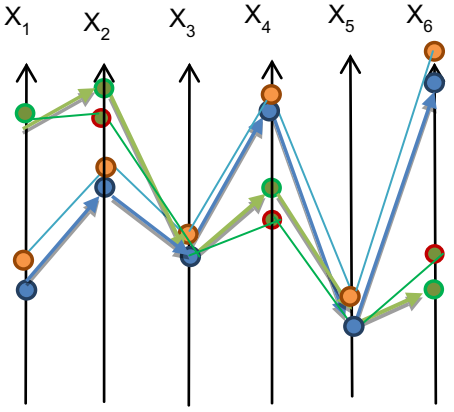



Fig. 16. Four 6-D points of two classes in Parallel Coordinates.

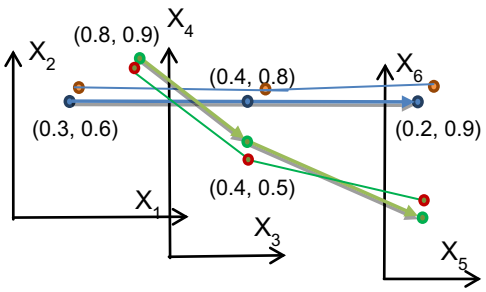



Fig. 17. Four 6-D points of two classes in Shifted Paired Coordinates with crossing lines.

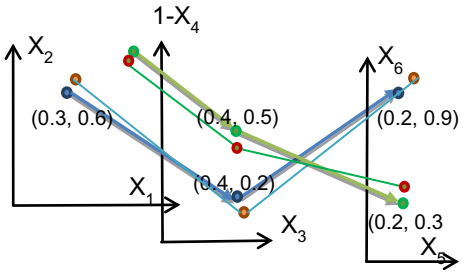



Fig. 18. Four 6-D points of two classes in Shifted Paired Coordinates with crossing lines with coordinate $X_4$ changed to 1-$X_4$ that removes crossing of lines caused by $X_4$ where: $A' = (0.3, 0.6, 0.4, \mathbf{0.2}, 0.2, 0.9)$ (thick blue line) and $C' = (0.8, 0.9, 0.4, \mathbf{0.5}, 0.2, 0.3)$ (thick green line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

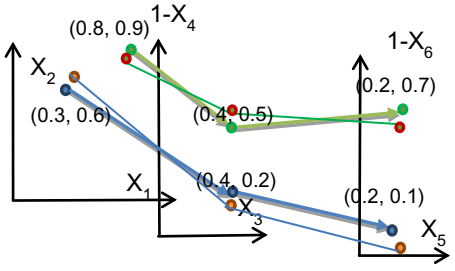



Fig. 19. Four 6-D points of two classes in Shifted Paired Coordinates with crossing of lines caused by $X_6$ eliminated by changing coordinate $X_6$ to 1-$X_6$ where: $A'' = (0.3, 0.6, 0.4, \mathbf{0.2}, 0.2, \mathbf{0.1})$ (thick blue line), $C'' = (0.8, 0.9, 0.4, \mathbf{0.5}, 0.2, \mathbf{0.7})$ (thick green line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In Fig. 20, the green line C became a preattentive horizontal straight line by shifting the coordinates ($X_1$, $X_2$) and ($X_5$, 1-$X_6$) down. Here the blue line is still not preattentive. In Fig. 21 it is made more preattentive by shifting pairs of coordinates ($X_1$, $X_2$) and ($X_5$, 1-$X_6$) horizontally to make the thick blue line a straight line.

In Fig. 21 it is made more preattentive by shifting pairs of coordinates ($X_1$, $X_2$) and ($X_5$, 1-$X_6$) horizontally to make the thick blue line a straight line.

Fig. 22 shows the next step of simplification where the green line C is collapsed to a single arrow between two points obtained by shifting pair of coordinates ($X_1$, $X_2$) to the right.

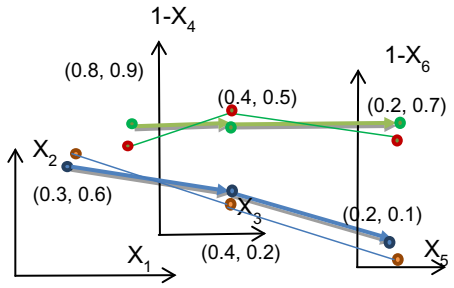



Fig. 20. Four 6-D points of two classes in Shifted Paired Coordinates without crossing lines and with monotone blue lines and horizontal thick green line obtained by shifting pairs of coordinates ($X_1$, $X_2$) and ($X_5$, 1-$X_6$) down to make the thick green line C horizontal. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

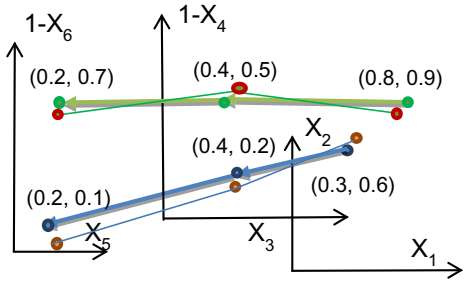



Fig. 21. Four 6-D points of two classes in Shifted Paired Coordinates without crossing lines and with horizontal thick green line and with a straight monotone thick blue line obtained by shifting pairs of coordinates ($X_1$, $X_2$) and ($X_5$, 1-$X_6$) horizontally to make the thick blue line a straight line. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

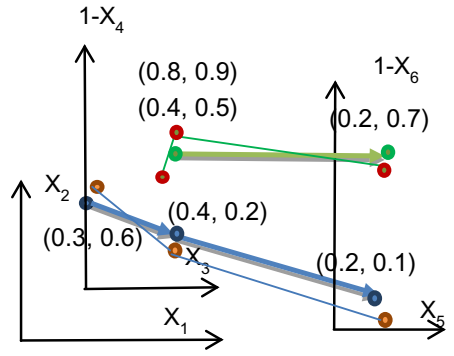



Fig. 22. Four 6-D points of two classes in Shifted Paired Coordinates without crossing lines and with a straight monotone thick blue line and horizontal thick green line obtained by shifting pair of coordinates ($X_1$, $X_2$) to the right to collapse C to a single arrow). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The final simplification step is presented in Fig. 23, where the green line C is collapsed into a single preattentive 2-D point shown as the green dot by shifting a pair of coordinates ($X_5$, 1-$X_6$) to the left. This representation remains reversible, i.e., all 6 values of each of these four 6-D points can be restored from these graphs.

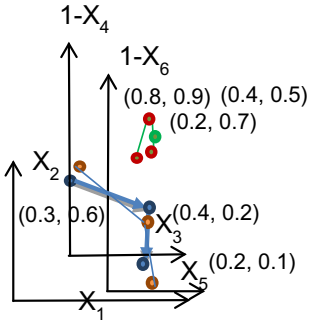



Fig. 23. Four 6-D points of two classes in Shifted Paired Coordinates without crossing lines obtained by shifting pair of coordinates ($X_5$, 1-$X_6$) to the left to collapse the thick green line C to a single point. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

## 8. Super-intelligence for high-dimensional data

The results presented above create an exciting opportunity for progress in the super-intelligence studies. While significant progress in Artificial Intelligence, Computational Intelligence, and Machine Leaning improved human abilities to discover the patterns in n-D data the direct human cognitive abilities to do this with a naked eye are extremely limited to relatively small 2-D and 3-D datasets.

Lifting this human cognitive limitation is in a drastic contrast with the opposite goal of reaching human-level machine intelligence for human abilities, which is the goal of other aspects of Artificial Intelligence, Computational Intelligence, and Cognitive Science. This opposite goal is deciphering the brain's existing cognitive abilities and mimicking human intelligence that use a naked eye very successfully to recognize and discover visual patterns, e.g., faces and facial expressions, in our physical 3-D world.

Thus we need both: (1) the deciphering of the brain and (2) the enhancing of it to be able to deal with abstract high-dimensional data as it is done with 2-D and 3-D data.

Compare it to building a machine that will fly as a bird. It is difficult to decipher the mechanism of bird flying. The history of aviation had shown that direct attempts to mimic it failed many times. Next, the machine that intends only to mimic a flying bird will be limited. It will not fly to the Moon and Planets. For flying that far a machine with *super-bird flying capabilities* is needed.

Similarly deciphering brain's ability to work visually with 2-D data hardly will give us a way to build a *super-intelligence* to deal with large *abstract n-D data*. This is a separate and very challenging task. Evolution has developed our brain in a particular form to adapt to a particular physical 3-D environment that did not include abstract high-dimensional data (n-D data) to be analyzed until the very recent *Big data era*.

This separate task requires the ideas beyond what is on the surface when humans solve their typical cognitive tasks in 2-D and 3-D. In the same way exploring how a bird is flying hardly will help in building a rocket to fly to the Moon that requires discovering more general flying principles. Similarly dealing with Big n-D requires discovering more general cognitive principles than we use for 2-D and 3-D data.

Is it always more difficult to discover more general principles than more specific ones? The history of the science tells us that it is not always the case. The modern flight theory that includes the propulsion theory and aerodynamics explains not only bird flight, but also rocket and aircraft flights. However this more general theory does not tell us anything about the physiology of bird flight at the level of muscles and the bird brain control of the flight. Thus higher generality does not mean abilities to explain all aspects of the bird flight. However it can help to discover and understand a mechanism of other related activities. For instance, the propulsion theory allows the understanding of an octopus motion. In our case it is discovering cognitive principles to deal with n-D data.

This brings us to the important point that for understanding some fundamental brain cognitive principles it is not necessary to study the brain itself first. Respectively to build such more general theory we can work on the task that brain does not support well, which is dealing with n-D abstract data. The goal is to understand and enhance brain's capability to deal with such n-D data. It includes experiments with the same n-D data where a human may or may not recognize the pattern depending on 2-D lossless representation of these n-D data. These experiments can tell about human abstract pattern recognition abilities providing data to build a cognitive pattern recognition/discrimination model.

After a discrimination model is built, the next question is: "What is the mental process in the brain behind this ability or inability?" The common approach in such tasks is collecting and analyzing the functional MRI data when the task is solved by subjects. In Murray, Kersten, Olshausen, Schrater, and Woods (2002) functional MRI was used to measure activity in a higher object processing area, the lateral occipital complex, and in primary visual cortex in response to visual elements that were either grouped into objects or randomly arranged. These authors observed significant activity increases in the lateral occipital complex and concurrent reductions of activity in primary visual cortex when elements formed coherent shapes.

Based on this observation they suggested that activity in early visual areas is reduced as a result of grouping processes performed in higher areas. These findings were used as an evidence for the brain predictive coding models of vision (Mumford, 1992; Rao & Ballard, 1999) that postulate that inferences of high-level areas are subtracted from incoming sensory information in lower areas through cortical feedback. Note that this study was conducted for 2-D and 3-D shapes such as shown in Fig. 24 without any relation to n-D data.

The predictive coding models of vision represent one side of two fundamental alternatives: local and distributed representation models/hypotheses for the brain to be

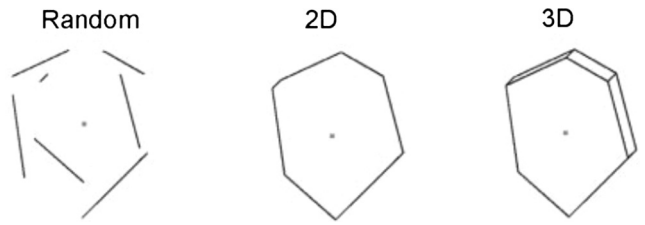



Fig. 24. Examples of different stimulus conditions (Murray et al., 2002).

biologically-adequate representations for observed high-level structures and cognitively-adequate models. There are several distributed representation cognitive models with bottom-up and top-down signals (Carpenter & Grossberg, 2016; Mumford, 1992) including the dynamic logic model that we advocate (Kovalerchuk, Perlovsky, & Wheeler, 2012) because of its ability to overcome combinatorial complexity. On the other hand while current deep leaning large Neural Networks may not be biologically-adequate their applied results are impressive.

## 9. Conclusion and future studies

This paper presented the concept of lossless visualization of n-D data as cognitive enhancer for discovering n-D data patterns. It includes definitions, algorithms and mathematical statements that demonstrate how n-D data representations in various general line coordinates simplify representation of n-D data in 2-D for better perceptual and cognitive abilities for visual pattern discovery. The advantages of GLC sub-class of Collocated Coordinates have been shown on real-world data and demonstrated on visual pattern simplification. The link with super-intelligence challenges has been established too.

The future studies are twofold: (1) enchasing methods for lossless representation of n-D data in 2-D, (2) clarifying brain cognitive processes associated with analysis of abstract n-D data. For the second issue future studies include gaze analysis: when humans analyze visual representations of abstract n-D data and discover n-D patterns. While eyes provide initial input of such visual information, visual perception, and cognition deeply involve the brain. Therefore the gaze analysis will help to look deeper into this complex process.

Combining eye tracking methodology, mathematical models from different fields, and the behavioral information which emerges in the analysis of n-D data will be a source of new knowledge of the cognitive processes. This will include future experiments that compare observers' performance in discovering n-D data patterns by analyzing 2-D graphs as a function of their fixations and simulations by the computations of these fixations.

These future studies will also help (a) to reveal the individual variability among the people in their perceptual and cognitive abilities for recognizing the abstract forms, and (b) to understand the visual and cognitive perception along with improving the accuracy, increasing efficiency, and decreasing the cost of n-D data analysis.

## References

Bertini, E., Tatu, A., & Keim, D. (2011). Quality metrics in high-dimensional data visualization: An overview and systematization. *IEEE Transactions on Visualization and Computer Graphics, 17*(12), 2203–2212.

Carpenter, G. A., & Grossberg, S. (2016). Adaptive resonance theory. In C. Sammut & G. Webb (Eds.), *Encyclopedia of machine learning and data mining*. Berlin: Springer-Verlag. http://dx.doi.org/10.1007/978-1-4899-7502-7 6-1.

Grishin, V., & Kovalerchuk, B. (2014). Multidimensional collaborative lossless visualization: Experimental study. In Luo (Ed.), *CDVE 2014, Seattle, Sept 2014. CDVE 2014, LNCS 8683* (pp. 27–35). Springer.

Hibbard, B. (2002). Super-intelligent machines. *Kluwer*.

Hoffman, P., & Grinstein, G. (2002). Survey of visualizations for high-dimensional data mining. In U. Fayyad, A. Wierse, G. Grinstein (Eds.), *Information visualization in data mining and knowledge discovery* (pp. 44–82). Academic Press.

Inselberg, A. (2009). *Parallel coordinates: Visual multidimensional geometry and its applications*. Springer.

Kandogan, E. (2001). Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In *Proc. 7th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 107–116).

Kovalerchuk, B. (2014). Visualization of multidimensional data with collocated paired coordinates and general line coordinates. In *Proc. SPIE 9017, visualization and data analysis* (p. 90170I).

Kovalerchuk, B., & Grishin, V. (2014). Collaborative lossless visualization of n-D data by collocated paired coordinates. In Y. Luo (Ed.), *CDVE 2014, Seattle, Sept 2014. CDVE 2014, LNCS 8683* (pp. 19–26). Switzerland: Springer.

Kovalerchuk, B., Perlovsky, L., & Wheeler, G. (2012). Modeling of phenomena and dynamic logic of phenomena. *Journal of Applied Non-classical Logics, 22*(1), 51–82.

Machine Learning Repository (2017). UCI. <https://archive.ics.uci.edu/ml/datasets/Iris>. <https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center>.

Michelucci, P., & Dickinson, J. L. (2015). The power of crowds. *Science, 351*(6268), 32. http://dx.doi.org/10.1126/science.aad6499.

Mumford, D. (1992). On the computational architecture of the neocortex. II. The role of cortico-cortical loops. *Biological Cybernetics, 66*, 241–251.

Munzner, T. (2015). *Visualization analysis and design*. CRC Press.

Murray, S., Kersten, D., Olshausen, B., Schrater, P., & Woods, D. (2002). Shape perception reduces activity in human primary visual cortex. In PNAS, Nov. 12, 2002 (vol. 99, no. 23, pp. 15164–15169). <www.pnas.orgycgiydoiy10.1073ypnas.192579399>.

Rao, R. P., & Ballard, D. H. (1999). Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience, 2*, 79–87.

Simov, S., Bohlen, M., & Mazeika, A. (Eds.). (2008). *Visual data mining*. Springer.

Superintelligence. Wikipedia, 11/12/2015. <https://en.wikipedia.org/wiki/Superintelligence>.

Tergan, S., & Keller, T. (Eds.). (2005). *Knowledge and information visualization*. Springer.

Ward, M., Grinstein, G., & Keim, D. (2010). *Interactive data visualization: Foundations, techniques, and applications*. Natick, MA: A K Peters, Ltd..

Wong, P., & Bergeron, R. (1997). 30 Years of multidimensional multivariate visualization. In G. M. Nielson, H. Hagan, & H. Muller (Eds.), *Scientific visualization – Overviews, methodologies and techniques* (pp. 3–33). IEEE Computer Society Press.