Arbitrary Dimensional Data

Frank Edughom Ekpar

Abstract—It is a well-known fact that numerous issues in many fields of human endeavor including, but not limited to, science and engineering, medicine, law enforcement and security, economics and finance, governance, psychology, philosophy, religion, and many other fields require the management of arbitrary dimensional data. However, systems permitting direct and efficient management of arbitrary dimensional data currently do not exist. In fact, contemporary systems such as graphical user interfaces for the management of data typically lack even the very concept of arbitrary dimensionality – failing to provide any practical way or means of managing arbitrary dimensional data. Here, we establish the foundational principles for a system permitting practical, direct and efficient management of arbitrary dimensional data. Furthermore, we demonstrate the effectiveness of our system by applying it to an experiment involving eight-dimensional (8D) medical and scientific data sets. Our system has immediate, far-reaching implications for numerous fields of human endeavor – enabling hitherto impossible solutions and applications and leading to deeper insights and improved understanding of numerous issues.

Index Terms—Arbitrary Dimensional Data, Multidimensional Data, Efficient Data Management, Intuitive Data Management, 3D Graphical User Interface, Big Data Representation, Dynamic View Prediction, Representative Matrix Formulation.

I. INTRODUCTION

Natural systems and even social systems that are often derived - at least in part - from or influenced by natural systems are inherently complex. And with the increasing sophistication of the technological tools at the disposal of human society, many artificial systems already boast of complexity levels approaching those of natural and social systems and are poised to attain even higher levels of complexity. This complexity demands the use of a large and ever increasing set of significant parameters for the adequate characterization and analysis of the underlying systems. Generally, each such significant parameter translates into a separate dimension in the data associated with the underlying systems. Consequently, important issues in every field of human endeavor including, but not limited to, science and engineering, medicine, law enforcement and security, economics and finance, governance, psychology, philosophy, religion, and many other fields require the management of arbitrary dimensional data. For example, the data associated with the dimensions encountered in string field theory as formulated by Michio K., Igor R. K., Nima A.[1-10] and other contributors in the fascinating quest for a unified, final theory of Physics are typically multidimensional – routinely approaching up to twenty-six (26) or more dimensions. The simplified two-dimensional string theory is formulated as a sum over surfaces embedded in one dimension that can be characterized and analyzed using the matrix model approach. Although Igor R. K.[2] notes that the matrix model approach has truly revolutionized the two-dimensional Euclidean quantum gravity, Michio K.[1] observes that this simple and powerful tool is severely limited by the fact that all string degrees of freedom are missing and consequently, many of the successes of the theory are intuitively difficult to interpret in terms of string degrees of freedom. Furthermore, features such as the discreet state and the w(∞) algebra arise in a rather obscure fashion. Michio K.[1] further observes that although Liouville theory explicitly encompasses all string degrees of freedom, it is notoriously difficult to solve, even for the free case. Nima A. et. al.[10, 11] have proposed a dual formulation for the S Matrix of N = 4 SYM as well as a new framework for solving the hierarchy problem that does not rely on either supersymmetry or technicolor but the critical issue of the lack of a direct, intuitive and efficient way to represent the dimensions in String Field theory and many other fields of human endeavor has not yet been resolved by any researcher. In mathematics, there is as yet no direct and intuitive representation of the dimensions in infinite-dimensional Lie algebras and Lie groups\(^2\). K. J. Schrenk et. al.[13] illustrate how a wide variety of apparently unrelated problems ranging from tracing water basins and river networks in landscapes[14-18] to the identification of cancerous cells in human tissues[19, 20] and the study of spatial competition in multispecies ecosystems[21, 22] as well as cracks or surviving paths through discretized maps possessing a universal fractal dimension that can be physically realized in terms of optimal paths under strong disorder[23-26], optimal path cracks[27, 28], loopless percolation[29, 30], or minimum spanning trees[31-36] can all be understood in terms of the same universal concept of fracturing a ranked surface and present results for higher dimensions. Like all the other researchers before them, K. J. Schrenk et. al.[12] fail to show how the higher dimensions involved in the wide range of problems with seemingly unrelated physical models that tumble into the same universality class can be represented directly and intuitively. Julie D. et. al.[37] describe a mental canvas – a tool for conceptual architectural design and analysis – that allows the designer to organize concept drawings in three dimensions (3D), and gradually fuse a series of possibly geometrically-inconsistent sketches into a set of 3D strokes. Predictably, the system introduced by Julie D. et. al.[37] is limited to three dimensions (and at most four dimensions with the addition of time-varying components) without facilitating direct, practical and efficient management of arbitrary dimensional data despite

Published on January 15, 2020.
F. E. Ekpak is with Admiralty University of Nigeria, IbUSA/Ogwashi-Uku Expressway, Delta State, Nigeria. (e-mail: frankekparng@gmail.com)

DOI: http://dx.doi.org/10.24018/ejers.2020.5.1.1703
the fact that such capabilities would make the system infinitely more useful. Frank E.[38] invented a system for creating and managing interactive virtual tours, that permits virtual tour elements to be disposed on arbitrarily sized and arbitrarily shaped surfaces. It is desirable to allow arbitrary dimensions for such interactive virtual tour elements. An excellent review of the state of the art in scientific and medical data visualization by Thomas W. et. al.[39] provides a comprehensive overview of the strengths and weaknesses of the best tools for scientific and medical data visualization. Without exception, all the systems reviewed by Thomas Walter et. al.[39] fail to provide any direct and practical means of managing arbitrary dimensional data despite the fact that virtually all the data considered is essentially arbitrary dimensional. The dire need for a simple and practical system for the direct and intuitive management of arbitrary dimensional data is clear from the conclusions reached through the excellent review of the state of the art in scientific and medical data visualization by Thomas W. et. al.[39] as well as the numerous examples provided in the foregoing.

In this paper, we address the problem of providing a practical system for the direct, efficient and intuitive management of arbitrary dimensional data by establishing the foundational principles for the system. The system permits the creation of one or more arbitrary dimensional instances of data that can be disposed on one or more arbitrarily-shaped and arbitrarily-sized surfaces and allows user interaction with the instances of the data. In this context, an instance is any presentation or rendering or display of any aspect of the data. Presentation of the data could be realized on literal display surfaces such as computer monitors or any other suitable display system as well as on conceptual surfaces representing computer memory, data abstractions or computer program-defined data structures, without limitation. Since a very large and growing subset of the data available to human society is amenable to processing or management on a computer or related systems which generally require interaction through a graphical user interface, we use the graphical user interface metaphor to elucidate the foundational principles of the system. However, it should be understood that any aspect of such graphical user interfaces should properly be interpreted as data and treated as an instance of the arbitrary dimensional data managed by the system. We show how new instances of arbitrary dimensional data can be synthesized, created or computed to facilitate seamless navigation of the data managed by the system. Our system dramatically simplifies the management of arbitrary dimensional data by reducing the addition of a new dimension to the addition of a row to a representative matrix and the removal of an existing dimension to the removal or deactivation of a row within a representative matrix. Finally, we demonstrate the effectiveness of the system by applying it to a thought experiment involving eight-dimensional (8D) medical and scientific data sets and introduce a general framework for the efficient management of the very large data sets that could be encountered in the application of the system in resource-constrained environments such as the Internet.

A survey of the relevant literature has been provided by the introductory Section I. The remainder of this article is organized as follows: Section II, the problem definition, methodology/approach is presented in the form of the description of the representative matrix formulations used to articulate the core principles of our solution. In Section III, we illustrate how new instances of arbitrary dimensional data could be synthesized or computed. Results in the form of the experiment involving eight-dimensional (8D) medical and scientific data sets and demonstrating the effectiveness of our system appear in Section IV. Section V discusses further results via the management of the very large data sets often associated with contemporary systems. Section VI provides concluding remarks and future scope.

To insert images in Word, position the cursor at the insertion point and either use Insert | Picture | From File or copy the image to the Windows clipboard and then Edit | Paste Special | Picture (with “Float over text” unchecked).

EUROPEAN JOURNAL OF ENGINEERING RESEARCH AND SCIENCE reserves the right to do the final formatting of your paper.

II. PROBLEM DEFINITION/METHODOLOGY/APPROACH: CORE PRINCIPLES: REPRESENTATIVE MATRIX FORMULATIONS

We start with any conventional instance of data such as an instance of a conventional one-dimensional (1D), two-dimensional (2D), three-dimensional (3D) or four-dimensional (4D) graphical user interface and apply a 2D, 3D or higher-dimensional representative matrix formulation of the core principles of our system to transform the conventional data instance into an arbitrary dimensional data ($\alpha$ – dimensional) instance. This process illustrates the ease with which existing limited data management systems could readily be transformed into more useful arbitrary dimensional data management systems based on the foundational principles of our system.

A. Two-Dimensional (2D) Representative Matrix Formulation

Fig. 1 illustrates the use of the representative 2D matrix formulation to present user interface instances. The instances of the user interface are indicated generally as

$$UI_{11}, UI_{12}, UI_{21}, UI_{22}, UI_{1j}, UI_{i1}, \ldots, UI_{ij}$$

and are disposed on an arbitrarily-shaped and arbitrarily-sized surface. The surface is depicted by the bounding box for the user interface instances in Fig. 1. The individual user interface instances are shown in a manner reminiscent of entries in a 2-dimensional (2D) matrix. Suffixes $i$ and $j$ (as in $UI_{ij}$) refer to row and column numbers within the 2D matrix.

Hence, $UI_{11}$ is the instance in row 1, column 1; $UI_{12}$ is the instance in row 1, column 2;
$\text{UI}_{i1}$ is the instance in row 2, column 1;
$\text{UI}_{i2}$ is the instance in row 2, column 2;
$\text{UI}_{ij}$ is the instance in row 1, column $j$, where $j$ is an arbitrary number,
$\text{UI}_{i1}$ is the instance in row $i$, column 1, where $i$ is an arbitrary number and
$\text{UI}_{ij}$ is the instance in row $i$, column $j$, where $i$ and $j$ are arbitrary numbers.

Fig. 1. User interface instances disposed on an arbitrarily-shaped and arbitrarily-sized surface using a 2D representative matrix formulation.

Entries in columns refer to specific instances of the user interface. Please note that the underlying user interface elements need not be distinct for each individual instance. For example, one or more instances may share the same user interface elements but may be used to present or display different views or perspectives or representations of data presented or displayed using the user interface. Thus, one or more entries in columns could represent different representations of data while sharing some or all underlying user interface elements.

Rows within the 2D matrix represent individual or separate dimensions. Accordingly, row 1 could be used to represent a new or additional dimension for the user interface instances contained in the columns within row 1, namely

$\text{UI}_{11}$, $\text{UI}_{12}$, $\text{UI}_{13}$, \ldots, $\text{UI}_{1j-1}$, $\text{UI}_{1j}$.

Hence, irrespective of the actual number of dimensions contained within a specific user interface instance such as $\text{UI}_{11}$ within row 1, the inclusion of $\text{UI}_{11}$ as a column (column 1 in this case) within row 1 increases the number of dimensions for the specific user interface instance and its derivative representations by 1. Hence if $\text{UI}_{11}$ was originally a conventional 2D user interface, then the inclusion of $\text{UI}_{11}$ as a column within row 1 increases the number of dimensions for $\text{UI}_{11}$ and its derivative representations by 1 and thus promotes $\text{UI}_{11}$ and its derivatives to a 3D user interface. A derivative instance in this context refers to any new instance of the user interface (such as $\text{UI}_{12}$) that shares some or all underlying user interface elements and that could be used to present or display a different view or perspective or representation of data presented or displayed using the user interface. Note that the concept of derivation is not limited to the sharing of one or more user interface elements but could be defined in terms of associations between instances. That is, a new instance could be considered a derivative of an existing instance if the two instances are associated with the same dimension (for example by being disposed within the same row of the representative matrix) within the user interface irrespective of whether the instances actually share any common elements or not. So entries within row 1 could be used to represent or display or present derivatives of $\text{UI}_{11}$ and thus promote $\text{UI}_{11}$ and derivatives of $\text{UI}_{11}$ to a 3D user interface in the case where $\text{UI}_{11}$ (and its derivatives) were originally conventional 2D user interface instances.

Adding a new dimension to the user interface simply involves adding a new row to the 2D matrix representing the user interface and representing, displaying, presenting or rendering derivative instances within the new row. Similarly, removing or disabling an unused or unwanted dimension from the user interface involves merely removing or disabling the affected row within the representative 2D matrix and disabling the representation, display, presentation or rendering of instances within the affected row.

Consequently, the 2D matrix representation of the user interface depicted in Fig. 1 and described in the foregoing is sufficient to permit the user interface to manage arbitrary dimensional data without limitation. The user simply adds a new row to the 2D matrix in order to add a new or additional dimension to the user interface.

Note that according to the foundational principles introduced herein, the individual user interface instances are themselves arbitrary dimensional. That is, each instance could be managed using a representative 2D matrix as described for the entire user interface.

One of ordinary skill in the art will readily appreciate that the actual representations of the instances -- which could be in the form of digital images or vector graphics on computer systems or any other suitable representation -- could be rendered or displayed or presented in a wide variety of ways or using a wide variety of means. For example, each instance could be displayed as a separate window within the desktop of a typical window-based operating system such as those available from Microsoft (Microsoft Windows, Windows Mobile, and so on), Apple (Apple OS, iPhone OS, iPad OS, and so on), Google (Android, and so on) or any other window-based operating system. Given a sufficiently large or high resolution display device or display surface, it is possible to display each instance as a region within a single window within a typical window-based operating system.

DOI: [http://dx.doi.org/10.24018/ejers.2020.5.1.1703](http://dx.doi.org/10.24018/ejers.2020.5.1.1703)
system desktop or similar surface. The exact configuration or arrangement of instances within such a window or the exact configuration or arrangement of windows (in the case where separate windows are assigned to separate instances) could be determined in accordance with user preferences or other relevant factors such as resource constraints. Suitable systems for managing the representations of the instances could also be adopted. For example, window swapping techniques allowing one or more instances to be viewed at a time or in a specific situation or providing means of selecting which specific instance to view at a specific time or in a specific situation could be adopted in resource-constrained environments. Alternatively, each instance could be displayed on a separate monitor or display device where the monitors or display devices are spread over any chosen geographical territory in any chosen spatial configuration or arrangement. One simple spatial arrangement could involve disposing the monitors or display devices in a manner reminiscent of a 2D matrix to closely match the 2D matrix representation of the user interface.

It should be apparent to one of ordinary skill in the art that the arbitrarily-sized and arbitrarily-shaped surfaces on which the user interface instances could be disposed could be 1-dimensional (constraining the instances to a linear or 1D configuration), 2-dimensional (constraining the instances to a planar or 2D configuration), 3-dimensional (constraining the instances to the familiar physical 3D spatial environment or configuration), 4-dimensional (for example comprising a spatial 3D configuration and a linear or 1D time dimension as in time-varying spatial 3D configurations or arrangements) or arbitrary dimensional by adopting the foundational principles introduced herein and possibly utilizing the 2D matrix paradigm described in this paper. Furthermore, it should be understood that the surfaces need not be literal or physical surfaces but could represent computer memory, rendering surfaces – such as device contexts in Microsoft Windows Software Development Kit parlance – or any other suitable representations of surfaces on which instances could be disposed.

Any of the well-known viewport and window management techniques including, but not limited to, scrolling, panning, and so on, could be employed where appropriate to facilitate the display or presentation of the instances.

The conceptualization of the user interface as a 2D matrix of arbitrary dimensional instances permitting new or additional dimensions to be added simply by adding new rows to the representative 2D matrix and permitting unused or unwanted dimensions to be removed or disabled simply by removing or disabling rows within the representative 2D matrix makes user interfaces enabled by the foundational principles introduced herein amenable to straightforward mathematical characterization and analysis.

Using the 2D matrix paradigm, useful characteristics associated with the user interface such as performance metrics could easily be generated and managed using well known and widely used mathematical tools. For example, each instance could be configured to collect and store - or transmit for further analysis – user data associated with any selected aspect of the user experience and this data could be represented and analyzed using the 2D matrix notation – the representation of the user interface itself as a 2D matrix of instances making the collection, analysis and utilization of the data easier or more efficient.

Performance metrics analysis could be facilitated by carrying out automatic, semi-automatic or manual collection and analysis of relevant user experience data. Automatic data collection could be implemented by any means or method that allows the tracking of relevant user actions such as mouse clicks, button activation, hits and misses on buttons, sliders or any other tracked user interface elements. Such tracking of using actions could readily be programmed into the user interface by one of ordinary skill in the art using readily available programming languages such as C, C++, JAVA, Python, HTML, HTML5, VRML, and so on in combination with suitable programming tools such as software development kits for any chosen computer system or application environment. Other means of tracking the user experience such as the use of suitably configured brain-computer interfaces or similar bio-technology systems or any other suitable systems to determine the level of excitement or frustration caused by the use of specific instances of the user interface or the amount of effort required by specific groups of users to master the user interface to a specific level of proficiency could also be utilized in the automatic collection of user experience data.

Semi-automatic data collection could involve the use of any of the automatic data collection methods described earlier (or any other suitable automatic data collection method) augmented by manual inspection and/or correction of errors in the automatically collected data.

Manual data collection could involve the explicit or implicit (where appropriate) use of user surveys, polls or queries to document or track the user experience.

Once user data associated with any specific instances has been collected, known matrix analysis methods or any other suitable mathematical tools could be used to compute and characterize performance metrics with a view to improving the user experience and building better – more responsive and more efficient – user interfaces.

Any other aspect of the user interface – such as design or presentation issues - could also be analyzed using matrix notation aided by the conceptualization of the user interface as a 2D matrix of arbitrary dimensional instances.

B. Three-Dimensional (3D) and Higher-Dimensional Representative Matrix Formulations

While it is very clear from the foregoing that the conceptualization of the user interface as a 2D matrix of arbitrary dimensional instances is sufficient to permit any arbitrary dimensional application of the user interface, some users may prefer the more familiar 3D spatial configuration. Accordingly, Fig. 2 shows how instances of user interfaces could be disposed on one or more arbitrarily-sized and arbitrarily-shaped surfaces along the lines of the familiar 3D spatial configuration.

DOI: http://dx.doi.org/10.24018/ejers.2020.5.1.1703

EJERS, European Journal of Engineering Research and Science
Vol. 5, No. 1, January 2020

49
In Fig. 2, three separate surfaces are shown as an example. Any number of surfaces could be used in practice. Each surface is arbitrarily-sized and arbitrarily-shaped and based on the foundational principles introduced herein, is arbitrary dimensional as already explained for the 2D matrix notation. The instances disposed on the surfaces are labeled

\[
\{UI_{11}\}, \{UI_{1j}\}, \{UI_{i1}\}, \{UI_{ij}\}
\]

for the first surface;

\[
\{UI_{11}\}_2, \{UI_{1j}\}_2, \{UI_{i1}\}_2, \{UI_{ij}\}_2
\]

for the second surface; (note that

\[
\{UI_{i1}\}_2 \text{ and } \{UI_{ij}\}_2 \text{ are not visible in Fig. 2}
\]

\[
\{UI_{11}\}_l, \{UI_{1j}\}_l, \{UI_{i1}\}_l, \{UI_{ij}\}_l
\]

for the last surface numbered 1 in Fig. 2 where 1 is the surface number and can be chosen arbitrarily to satisfy the demands of a given application. In this conceptualization the surfaces labelled 1, 2, ..., 1 (where 1 is any number) constitute a new or additional dimension for the user interface.

By considering the collection of surfaces numbered 1, 2, ..., 1 as a new or additional dimension (separate from the familiar rows – numbered 1, 2, ..., i and columns – numbered 1, 2, ..., j of the 2D representation already described which together constitute an unlimited number of dimensions) for the representative matrix, the matrix could be extended to a 3-dimensional or 3D form. Just as explained for the 2D matrix formulation, techniques for the mathematical characterization and analysis of 3D matrices could be applied to this 3D matrix formulation for the representative matrix.

It is very clear from the foregoing description of the foundational principles for arbitrary dimensional data management that any graphical user interface including, but not limited to, contemporary user interfaces such as those available in widely used operating systems such as Microsoft Windows, Apple OS (iPhone OS, iPad OS, and so on), Google Android OS and so on, could readily be transformed into a much more versatile, useful, responsive and efficient arbitrary dimensional graphical user interface by applying the foundational principles established herein.

The unlimited (within normal resource constraints) number of dimensions permitted by the foundational principles established herein allow new and useful relationships in data to be discovered that could not be discovered or that could only be discovered by exerting an inordinate amount of effort or expending an inordinate amount of resources using the prior art.

Entirely new classes of applications that were hitherto impossible to realize or that could only be realized through the expenditure of inordinate amounts of effort and resources could easily and directly be realized on the basis of the foundational principles established herein.

III. SYNTHESIZING NEW INSTANCES OF ARBITRARY DIMENSIONAL DATA

New instances of the user interface could be synthesized or created or computed by utilizing existing instances of the user interface. Interpolation techniques requiring neighborhood relationships or associations between the existing instance or instances and the new or synthesized instances could be employed. Such interpolation techniques include, but are not limited to, bilinear interpolation (especially for 2D matrix representations), cubic interpolation, spline interpolation, tri-linear interpolation (especially for 3D matrix formulations), and so on.

The synthesis of new instances could be viewed as a means of improving the resolution of the user interface by permitting the creation or synthesis or prediction of new instances that may improve the value of the user interface.

One of ordinary skill in the art would appreciate that interpolation is by no means the only way to improve resolution or to synthesize, create or predict new instances. New instances could be synthesized without reference to any existing instances. Physical constraints or other relevant factors within a given application could allow new instances to be predicted, created or synthesized without reference to existing instances.

Furthermore, synthesis of new instances on the basis of one or more existing instances could be accomplished using any other suitable means apart from the interpolation techniques mentioned in the foregoing.

When utilizing a 3D matrix formulation for the
representation of the user interface, tri-linear interpolation could be used to synthesize or compute new instances of the user interface by applying the data from neighboring instances.

Figure 3 depicts tri-linear interpolation for a user interface instance bounded by eight neighboring user interface instances in a 3-dimensional spatial configuration.

Figure 3. Demonstration of tri-linear interpolation for a user interface instance bounded by eight neighboring user interface instances in a 3-dimensional spatial configuration.

T(x, y, z) = (V000*(1-x) + V100*x) * (1-y) + ((V010*(1-x)+V110*x)*y) *(1-z) + V001*(1-x)+V101*x)*(1-y)+((V011*(1-x)+V111*x)*y)*z

Note that L0, L1, L2 and B0, B1 are intersection points between adjacent faces of the cube formed by the neighboring points (V000, V100, V101, V001, V101, V010, V110, V111 and V011) for which data is available and the target point – T(x, y, z) - for which no data is available.

It should be obvious to one of ordinary skill in the art that the correct interpretation of the entities represented by the data points labeled V000, V100, V101, V001, V101, V010, V110, V111 and V011 for which data is available depends on the specific application. For instance, data within an instance could be represented as an image with elements comprising pixels or fractions of pixels (for sub-pixel precision) with the typical red, blue, green, alpha (RGBA) quad values for the associated red, green and blue color channels and alpha transparency channel and associated 3D coordinates x, y, z, and an optional time component that could be used in the case of video or time-varying image data. In this case the foregoing equation for tri-linear interpolation could be interpreted as a pixel-wise operator permitting the synthesis of new pixels for new instances by utilizing the values of existing pixels for existing instances.

IV. EXPERIMENT AND DISCUSSION: EIGHT-DIMENSIONAL (8D) MEDICAL AND SCIENTIFIC DATA SET APPLICATION

Some of the advantages of the foundational principles established herein over contemporary systems will be demonstrated via the application of the resulting system to the management of data in the medical and scientific field.

The data comprised 8-dimensional or 8D representations of miscellaneous colonies of bacteria that were studied under a microscope to monitor their development under a specific set of conditions. Some species in the colonies exhibited luminescence or emitted detectable light when exposed to a specific chemical or reagent. 3-dimensional or 3D video micrographs effectively comprising 4-dimensional or 4D image data were captured under controlled conditions. Each video was of the same duration. Thus, the primary data for the example is the 4D image data (referred to as 4D_IMAGE for the purpose of this illustration) that the 3D video micrographs represent. One of the parameters used in the capture of the data was the wavelength of the light used. Three separate wavelengths representing the Red, Blue and Green colors on the color spectrum were used. This constituted the fifth dimension (referred to as WAVELENGTH for the purpose of this illustration) for the data.

Under the WAVELENGTH dimension, the original population of the colonies at the beginning of each recording could be represented. The sixth dimension for the data was the environment (labeled ENVIRONMENT for the purpose of this illustration) under which the bacteria were monitored. Two separate environments were utilized. In one environment, the bacteria were exposed to a hostile fungus that caused a certain portion of the colony to die off. In the other environment, no hostile fungus was present and conditions were favorable for the development of the colonies. The ENVIRONMENT dimension could be adapted to represent the percentage change in the population at the end of the recording as compared with the original population at the beginning of the recording. As mentioned earlier, some species of bacteria in the colonies exhibited photoluminescence in the presence of a certain chemical or reagent. The presence or absence of this photoluminescence-causing chemical or reagent and the measurable effect produced (labeled PHOTOLUMINESCENCE for the purpose of this illustration) constituted the seventh dimension for the data. Video was captured for a week every other day of the week beginning on a Monday. Hence, data was available for Monday, Wednesday, Friday and Sunday. There was no data for Tuesday, Thursday and Saturday. The day of the week (labeled DAY for the purpose of this illustration) on which data was captured represented the eighth dimension for the data.

3D matrix representation has been adopted for the management of the user interface instances for this 8D medical and scientific application.

The first four dimensions are tied up with the 3D video data – constituting a 4D image data set (referred to as 4D_IMAGE for the purpose of this illustration) with pixel elements.
For the fifth dimension (the first row or row 1 of the 2D face of the 3D representative matrix), the wavelength of the light at which the 3D video data was captured has been utilized. This dimension is labeled WAVELENGTH for the purpose of this illustration and permits three separate values, namely RED (for the red light wavelength), GREEN (for the green light wavelength) and BLUE (for the blue light wavelength). Instances associated with the WAVELENGTH variable could be configured to display the 3D video data as captured under RED, GREEN or BLUE light in the appropriate columns.

The sixth dimension (the second row or row 2 of the 2D face of the 3D representative matrix) is tied up with the environment (labeled ENVIRONMENT for the purpose of this illustration) under which the bacterial colonies develop. FAVORABLE is a value of the ENVIRONMENT parameter indicating a favorable environment marked by the absence of hostile fungi while HOSTILE is a value of the ENVIRONMENT parameter indicating a hostile environment marked by the presence of hostile fungi. Instances associated with the ENVIRONMENT parameter could be configured to display only those bacterial populations that remain alive under HOSTILE or FAVORABLE environmental conditions as well as information – possibly in the form of text or textual elements – indicating the percentage of population of remaining bacteria compared with the original population of the colony. Note that for each instance or column entry within the sixth dimension (the second row or row 2 of the 2D face of the 3D representative matrix), the ENVIRONMENT information unique to the sixth dimension could be combined with the associated wavelength information from the fifth dimension. Thus, column entries within the second row on any surface on which instances are disposed could display the 3D video in RED, GREEN or BLUE light (depending on the column number – for example RED for column 1, GREEN for column 2 and BLUE for column 3) filtered to display only those bacterial populations that BOTH exhibit luminescence AND remain alive in the presence of a HOSTILE or FAVORABLE environment depending on the specific value of the ENVIRONMENT parameter specified for the affected instance.

From the foregoing, it should be very clear to one of ordinary skill in the art that this MEDICAL AND SCIENTIFIC application demonstrates that an arbitrary dimensional user interface created in accordance with the foundational principles established herein indeed permits any arbitrary dimensional data to be faithfully and directly represented by the instances of the user interface. As explained earlier, the cumulative effects of successive dimensions could be reflected faithfully and directly in any given instance. This is very clear from the faithful and direct representation of the cumulative effects of the fifth dimension (WAVELENGTH), the sixth dimension (ENVIRONMENT) and the seventh dimension (PHOTOLUMINESCENCE) on all affected column entries or user interface instances as shown here.

The day of the week (labeled DAY for the purpose of this illustration) on which data was captured constitutes the subject of the eighth dimension (the third dimension of the 3D representative matrix) and instances associated with this parameter could be disposed on a separate surface for a specific value of the parameter.

In this MEDICAL AND SCIENTIFIC application, data was available for Monday, Wednesday, Friday and Sunday.

Note that although data was not available for Tuesday, Thursday and Friday, the foundational principles established herein permit the synthesis of data or instances for each of these days for which data was not available. Furthermore, data could be synthesized for sub-day points, that is, for any hour or in fact for any time (even down to the millisecond or down to any arbitrary desired time precision, limited only by the precision of the hardware and/or software systems on which the system is implemented) whatsoever between the first day – Monday – and the last day – Sunday – for which data was available.

Consequently, our system permits seamless navigation of the arbitrary dimensional data (8D data in this specific application) even for points or days for which data was not available or for which data was not collected or captured. The synthesis or creation of instances for Tuesday on the basis of the foundational principles introduced herein will be illustrated shortly. For this application, the 2D matrix row values and their associated parameters and actual data dimensions are as follows:

\[
\begin{align*}
&i = 1 \text{ (WAVELENGTH \{RED, GREEN, BLUE\})} \text{ [FIFTH DIMENSION]} \\
&i = 2 \text{ (ENVIRONMENT \{HOSTILE, FAVORABLE\})} \text{ [SIXTH DIMENSION]} \\
&i = 3 \text{ (PHOTOLUMINESCENCE \{INTENSITY\})} \text{ [SEVENTH DIMENSION]} \\
\end{align*}
\]

The following tables show the associated column entries or instances for Monday, Wednesday, Friday and Sunday.
A. Interpretation of Instance Contents

```
| UI_11 | UI_12 | UI_13 |
|-------|-------|-------|
| Monday | Monday | Monday |
```

Each instance could contain user interface elements such as buttons, sliders, text, and so on, for data navigation (including 3D video navigation in this example) and for page flipping to enable display of data for other days (in fact for any arbitrarily selected day, time or point between the first and last days for which data was collected) apart from the specific day of the week which the instance represents. This feature could be made common to all instances. In particular, user interface elements enabling arbitrary selection of data for any desired day of the week for display could be provided in each instance.

We can summarize in a table with illustrative descriptions the contents of the column entries or instances for Monday. Data for other days for which data was available or for which data was collected or captured such as Wednesday, Friday and Sunday could be interpolated in a similar fashion.

V. MANAGING VERY LARGE DATA SETS

One common way to represent data associated with user interface instances is to interpret the data as an image comprising groups of pixels or groups of fractions of pixels for sub-pixel precision. Such pixels could be interpreted as representing suitably formatted colors such as the typical red-blue-green-alpha (RGBA) color quad as well as associated 2D or 3D coordinates. Other interpretations suited to different applications are possible.

It should be apparent to one of ordinary skill in the art that the data associated with the arbitrary dimensional instances permitted by the foundational principles established herein could become very large and difficult to manage in resource-limited environments such as the present Internet.

Efficient data management techniques pertaining to such applications including the use of predictive loading of relevant data and possible subsequent presentation on a display window or computer monitor or any other suitable device or system based on a dynamic prediction of the user’s point of view within the data stream could be applied to enable practical implementation and acceptable performance of applications of the system for very large data sets and related applications on off-the-shelf personal computer systems. When data associated with any selected instance is characterized in the form of pixel values of images representing the data, dynamic view prediction of the user’s field of view and associated intelligent data management techniques could be used to facilitate efficient data navigation.

Generally, any data set including, but not limited to very large data sets or image data sets, could be navigated seamlessly even in resource-limited environments such as the Internet by presenting a user-selected region of interest and permitting real-time operation by dynamically predicting and making available subsets of the original data set.

Equations 1, 2, 3 and 4 summarize aspects of the general principles of operation for the seamless navigation of data sets including, but not limited to, very large data sets or image data sets.

\[ R^K = f(X^N) \] (1)

\[ X^N = \{x_1, x_2, ..., x_N\} \] (2)

\[ \{T, \theta, \varphi, r, \lambda\} \subseteq X^N \] (3)

\[ \{B_{11}, ..., B_{NM}\} \subseteq R^K \] (4)

In Equation 1, \( R^K \) denotes the region of interest within the original data set characterized by a set of \( K \) features. The features could be a set of pixels in a 2D or 3D still or time-varying image, a set of image blocks (possibly at a given level of a resolution-based hierarchical pyramid) within a 2D or 3D still or time-varying image, or any other suitable feature set within an appropriate data set. \( X^N \) is a set of \( N \) parameters (decomposed into the component parameters \( \{x_1, x_2, ..., x_N\} \) in Equation 2) that determine the characteristics of the region of interest. As indicated in Equation 1, the region of interest is a function of the parameter set \( X^N \).

Note that for sufficiently small data sets, the region of

DOI: http://dx.doi.org/10.24018/ejers.2020.5.1.1703
interest could encompass the entire data set.

For image data, the observation that interactive rendering of the image data set involves the display of a relatively small (compared to the size of the underlying image data itself) view window could be exploited to adopt a robust two-tier or bi-level representation of the image. The first level would contain a virtual view of an entire image frame as a single continuous set of pixels. Note that the actual pixels – which in reality could be representations of entities other than images – need not be associated with a single image or image frame but could be considered as a continuous collection or set for simplicity in management. The region of interest or view window could then be calculated using a suitable application-specific feature set such as the time-varying 3D spherical coordinate feature set depicted in Equation 3 where \( T \) represents time, \( \theta \) represents the pan angle, \( \phi \) represents the tilt angle or azimuth, \( r \) represents the radius of the sphere while \( \lambda \) represents a zoom level that could be used to control the amount of detail for each view – as a subset of the total parameter set \( \mathbf{X}^{\text{N}} \). Since a single image frame can be very large, it is generally impractical to attempt to load the entire image frame (typically corresponding to tens of gigabytes or even terabytes or more of physical memory for certain applications) into memory at once. Consequently, the second level could comprise a segmentation or partitioning of each image frame into distinct image blocks of a size and color depth that facilitates straightforward manipulation on an average personal computer. This partitioning scheme could be accomplished by segmenting each image frame into \( N \times M \) distinct image blocks labeled \( B_{11}, B_{12}, \ldots \)

As illustrated in Equation 4 where the set of image blocks is depicted as a subset of the region of interest. The size of each partition can be chosen such that the view window straddles just a couple of image blocks. In this case only those image blocks in the second layer that are covered or straddled by the view window need be loaded into memory for the manipulation or rendering of the view, leading to a significantly reduced memory footprint.

The use of a two-tier image representation scheme permits alternate views of the image data that make further manipulation easier. For example, the simplicity of the first level permits the application of a multi-resolution pyramid representation of the image data, such as that described by Peter J. Burt et al.,

for efficient compression, storage and transmission and optionally for adaptive rendering that maintains a constant frame rate. A thumbnail of the entire image could also be generated at the first level. Such a thumbnail could be used to display a lower resolution version of the view window while waiting for image data to be retrieved and/or decompressed. Furthermore, the dynamic view prediction and on-demand loading algorithms described hereinafter are readily applicable to the second tier’s image block representation.

Following is an outline of the process of visualizing the data sets based pre-processing of the data described in the foregoing. First, a view window representing the segment of the current image frame that is indicated by the view parameters is specified or computed. In a given implementation with 3D spherical coordinate representation, four view parameters such as the pan angle, the tilt angle or azimuth, the radius of the sphere and the zoom or scale factor could be used to control the view. Other relevant view parameters or factors could be considered as appropriate for any given application. User input could be received via the keyboard and/or mouse clicks within the view window. Suitable gesture recognition interfaces or touch-based interfaces or brain-computer interfaces or any other suitable interface could be used to receive input, provide feedback or generally enable user interaction. Alternatively, a head-mounted display and orientation sensing mechanism could also be used. Views could be generated based on view window size and received input. In order to facilitate interactive rendering without the latencies and other limitations associated with contemporary systems, the rate of change of each of the view parameters with respect to time could be computed dynamically. The computed rate of change could then be used to predict the value of any selected parameter at any desired time in the past or future.

The following equation illustrates the use of the dynamic view prediction algorithm for a specific view parameter \( P \) - which could be the pan, tilt, radius, zoom level, or any other suitable parameter.

\[
P = p_0 + KaT
\]

In the foregoing equation, \( P \) is the predicted value of the parameter at time \( T \), \( p_0 \) is the current value of the parameter, \( a \) is the dynamically computed rate of change of the parameter with respect to time and \( K \) is a scale factor, usually 1. The values of the parameters predicted by the foregoing equation could be used to determine which specific image blocks need to be loaded into memory at any given time.

A computer software implementation using a background thread dedicated to loading those image blocks that are covered by the current view as well as any additional image blocks that might be needed for rendering the view in the future or past, that is, a number of future or past time steps, could be used. The exact value or duration (or optimum value for applications in which variations in the time step are acceptable) of the time step depends on the requirements of a given application. While typical applications on a desktop computer used alone or in association with the current Internet might offer acceptable performance with time steps measured in milliseconds and generally less than one second or some other similar figure, other more demanding applications may require much shorter time steps. Much longer time steps in the order of seconds, minutes, hours or longer could be acceptable in less demanding or other unique applications. Since the number of image blocks per frame is usually small, it is practical to preload, pre-fetch or pre-synthesize as appropriate, image blocks that would be required for rendering one or more time steps ahead or behind the current presentation time or the current time step – permitting smooth rendering at real-

DOI: http://dx.doi.org/10.24018/ejers.2020.5.1.1703
time rates. Hence the prediction of the subset of data that would correspond to the selected region of interest one or more time steps separate from the current presentation time or the current time step as explained in the foregoing could be utilized to permit smooth and efficient navigation of very large data sets even on resource-limited systems. The image data could be distributed from a server over the Internet or other network or accessed from local storage on a host computer. Any other alternative source and method of distribution could be used where appropriate.

By providing access to the predicted data – or more generally subsets of the data, our system enables efficient navigation of the data.

One of ordinary skill in the art would appreciate that the predicted subsets of data could encompass the entire data set for sufficiently small data sets. There are no limitations on the sizes of the predicted data sets or subsets with respect to the whole data set.

The display or presentation of the predicted data sets could be the manner in which access is granted to the data or the manner in which the data is utilized. Any other suitable method or means of utilizing the data could be employed as appropriate to a given application. As explained previously, predicted data could be used to facilitate efficient distribution of the data set. Generally, any other uses of the data apart from visualization or distribution could be employed as appropriate for any given application without limitation.

It should be obvious to one of ordinary skill in the art that although data represented as images has been used to illustrate the concepts for very large data navigation according to the foundational principles established herein, the principles are not limited to image data but could be applied directly to any large data. For simplicity and convenience, the data could be converted or transformed into image format where appropriate and then converted or transformed back into the original format as demanded by specific applications of the system. Raw data in any other format apart from the image format could also be processed directly as illustrated for images to facilitate smooth and efficient navigation of the data even in resource-limited environments.

Studies with image visualization systems have consistently shown that the use of a damping or inertial function to facilitate gradual changes in view parameters leads to the perception by users of a vastly smoother, more natural and more intuitive viewing process. This observation can be exploited by the dynamic view prediction algorithm to provide smooth, interactive distribution and visualization of very large image data sets even over relatively slow network connections such as the current Internet and other bandwidth-limited scenarios without the latencies and other deficiencies associated with the prior art. The gradually changing view parameters would then permit many more future or past time steps to be computed, preloaded, pre-fetched and/or pre-synthesized as appropriate to a greater degree of accuracy.

It should be noted that while the foregoing schemes for the management of very large data sets from systems based on the foundational principles established herein could be appropriate in situations where memory, computing and associated resources are limited with respect to the amounts of data processing required for effective utilization of the systems, much simpler and more straightforward data management techniques could be employed in situations with fewer data points from the instances.

VI. CONCLUSION AND FUTURE SCOPE

We have established the foundational principles for a system for the practical, direct and efficient management of arbitrary dimensional data – dramatically simplifying the addition of a new dimension to any data set to the addition of a corresponding row and the removal of an existing dimension from any data set to the removal or deactivation of a corresponding row within a representative matrix formulation. Additionally, we have demonstrated the effectiveness of our system by applying it to a thought experiment involving eight-dimensional (8D) medical and scientific data sets. With the establishment of the foundational principles for the practical, direct and efficient management of arbitrary dimensional data, an exciting new era has dawned for the management of any kind of data with immediate, far-reaching implications and enabling hitherto impossible solutions and applications and leading to deeper insights and significantly improved understanding of numerous critically important issues in practically all fields of human endeavor.

REFERENCES

[1] Michio K. Sub-critical Closed String Field Theory in D Less Than 26. Phys. Rev. D 49, 5364-5376 (1994).
[2] Klebanov, I. R. String Theory in Two Dimensions. Proceedings of the Trieste Spring School 1991, eds. J. Harvey et. al. (World Scientific, Singapore, 1992).
[3] Ginsparg, P & Gregory, M. Lectures on 2D Gravity and 2D String Theory. JCTP, P23-92 (1992).
[4] Jevicki, A. Developments in 2D String Theory. BROWN-HET-918 (1993).
[5] Klebanov, I. R. & Polyakov, A. M. Interaction of Discrete States in Two-Dimensional String Theory. Mod. Phys. Lett. A6, 3273-3281 (1991).
[6] Gross, D. J. Klebanov, I. R. & Newman, M. J. The Two-Point Correlation Functions of the One Dimensional Matrix Model. Nucl. Phys. B350, 621–634 (1991).
[7] Ambjorn, A. & Jurkiewicz, J. Renormalization of 3d quantum gravity from matrix models. Phys. Letts. B581, 255-262 (2004).
[8] D’Hoker, E. Lecture Notes on 2D Quantum Gravity and Liouville Theory. UCLA/91/ITEP 35 (1991).
[9] Gubser, S. S., Klebanov, I. R. & Polyakov, A. M. Gauge Theory Correlators from Non-Critical String Theory. Phys. Lett. B428, 105-114 (1998).
[10] Arkani-Hamed, N., Cachazo, F., Cheung, C. & Kaplan, J. A Duality For The S Matrix. Journal of High Energy Physics (JHEP) 1003:020 (2010).
[11] Arkani-Hamed, N., Dimopoulos, S. & Dvali, G. The Hierarchy Problem and New Dimensions at a Millimeter. Phys. Lett. B429, 263-272 (1998).
[12] Kac, V. J. Simple Irreducible Graded Lie Algebras of Finite Growth. Math. USSR-Izvestija 2, 1271-1311 (1968).
[13] Schrenk, K. J., Araujo, N. A. M., Andrade Jr, J. S. & Herrmann, H. J. Fracturing ranked surfaces. Nature Scientific Reports 2, 348 (2012).
[14] Stark, C. P. An invasion percolation model of drainage network evolution. Nature 352, 423–425 (1991).
[15] Maritan, A., Colaiori, F., Flammini, A., Cieplak, M. & Banavar, J. R. Universality classes of optimal channel networks. Science 272, 984–986 (1996).
[16] Manna, S. S. & Subramanian, B. Quasirandom spanning tree model for the early river network. Phys. Rev. Lett. 76, 3460–3463 (1996).
sman, G., Rushmeier, H. E. & MacMillan, A. M. Local migration promotes competitive restraint in a host-pathogen ‘tragedy of the commons’. *Nature* 442, 75–78 (2006).

[22] Mathiesen, J., Mitarai, N., Sneppen, K. & Trusina, A. Ecosystems with mutually exclusive interactions self-organize to a state of high diversity. *Phys. Rev. Lett.* 107, 188101 (2011).

[23] Cieplak, M., Maritan, A. & Banavar, J. R. Optimal paths and domain walls in the strong disorder limit. *Phys. Rev. Lett.* 72, 2320–2323 (1994).

[24] Cieplak, M., Maritan, A. & Banavar, J. R. Invasion percolation and Eden growth: geometry and universality. *Phys. Rev. Lett.* 76, 3754–3757 (1996).

[25] Fehr, E. *et al.* New efficient methods for calculating watersheds. *J. Stat. Mech.* P09007 (2009).

[26] Fehr, E., Kadau, D., Andrade Jr, J. S. & Herrmann, H. J. Impact of perturbations on watersheds. *Phys. Rev. Lett.* 106, 048501 (2011).

[27] Andrade Jr, J. S., Oliveira, E. A., Moreira, A. A. & Herrmann, H. J. Fracturing the optimal paths. *Phys. Rev. Lett.* 103, 225503 (2009).

[28] Oliveira, E. A., Schrenk, K. J., Aratú, N. A. M., Herrmann, H. J. & Andrade Jr, J. S. Optimal-path cracks in correlated and uncorrelated lattices. *Phys. Rev. E* 83, 046113 (2011).

[29] Porto, M., Havlin, S., Schwarzer, S. & Bunde, A. Optimal path in strong disorder and shortest path in invasion percolation with trapping. *Phys. Rev. Lett.* 79, 4060–4062 (1997).

[30] Porto, M., Schwartz, N., Havlin, S. & Bunde, A. Optimal paths in disordered media: scaling of the crossover from self-similar to self-affine behavior. *Phys. Rev. E* 60, R2448–R2451 (1999).

[31] Barabási, A.-L. Invasion percolation and global optimization. *Phys. Rev. Lett.* 76, 3750–3753 (1996).

[32] Dobrin, R. & Duxbury, P. M. Minimum spanning trees on random networks. *Phys. Rev. Lett.* 86, 5076–5079 (2001).

[33] Jackson, T. S. & Read, N. Theory of minimum spanning trees. I. Mean-field theory and strongly disordered spin-glass model. *Phys. Rev.* E 81, 021130 (2010).

[34] Çifçi, K. Minimum spanning tree reflects the alterations of the default mode network during Alzheimer’s disease. *Ann. Biomed. Eng.* 39, 1493–1504 (2011).

[35] Goyal, S. & Puri, R. K. Formation of fragments in heavy-ion collisions using a modified clusterization method. *Phys. Rev. C* 83, 047601 (2011).

[36] Hubbe, M., Harvati, K. & Neves, W. Paleoamerican morphology in the context of European and East Asian Late Pleistocene variation: implications for human dispersion into the New World. *Am. J. Phys. Anthropol.* 144, 442–453 (2011).

[37] Dorsey, J., Xu, S., Smedresman, G., Rushmeier, H. E. & MacMillan, L. The Mental Canvas - A Tool for Conceptual Architectural Design and Analysis. *Proc. IEEE Conf. on Comp. Graphics and Applications*, 201-210 (2007).

[38] Ekpar, F. E. Method and Apparatus for Creating Interactive Virtual Tours. *United States Patent 7567274* (2009).

[39] Walter, T., Shattuck, D. W., Baldock, R., Bastin, M. E., Carpenter, A. E., Duze, S., Jan, E. Fraser, A., Hamilton, N., Pieper, S., Ragan, M. A., Schneider, J. E., Tomanec, P. & Héritéch, J. K. Visualization of image data from cells to organisms. *Nature Methods* 7, S26–S41 (2010).

[40] Burt, P. J. & Adelson, E. H. The Laplacian Pyramid as a Compact Image Code. *IEEE Trans. on Communications*, 532-540 (1983).