TOWARDS VISUAL-BANDWIDTH: GETTING CLOSE TO ONE'S EXPERIMENT DATA

Mark R. Titchener
The University of Auckland
Division of Science and Technology, Tamaki Campus, Morrin Road, Private Bag, Auckland

Keywords: information visualisation, graphical data interface, data animation.

Abstract: This paper presents a novel general purpose visualisation environment $\mathcal{S_P O_D}$ (Space Odyssey), that has evolved in relation to our research needs involving analysis and measurement of chaos and complexity in time-series data. More specifically our attention has turned to deriving sleep state information from EEG/EOG. $\mathcal{S_P O_D}$ operates largely as an interpreter of formated data files, to display surfaces, static and/or animated line and point graphs, individually or simultaneously in a virtual 3-D viewing space. It comfortably handles surfaces of more than 100,000 vertexes, and combinations of more than 15,000 static and/or animated graphs. User controls allow dynamic changes to viewing angle, lighting and display parameters. $\mathcal{S_P O_D}$ is ideally suited to 'scoping' experiment data and results, for visually debugging complex processing algorithms, or simply providing visual insight into complex data sets.

1 INTRODUCTION

This paper briefly introduces $\mathcal{S_P O_D}$ (2005: Space Odyssey), an interactive graphical display tool offering a combination of features tailored for scoping experiment data. The name is taken from the Kubrick film (Clarke and Kubrick, 1968), in which animated graphics featured with appealing simplicity as a backdrop to many of the scenes.

A plethora of graphics oriented applications already exist! Why another? Many applications, like Matlab (Moler, 1980), Mathematica, Maple (Char et al., 1983), S (Becker and Chambers, 1981), and more recently R (Ihaka and Gentleman, 1996), combine numerical and programming options with a broad range of graphical support features. Not all research projects sit comfortably within these self-contained computationally oriented environments.

Our own research in sleep related studies has evolved largely within the context of UNIX. The shell environment (Kochan and Wood, 1990), with its notion of standard I/O, filters, pipes, and redirection, encourages a distinctive mode of work, one that places a reliance on tools, rather than on applications. An established range of highly optimised *tools* are standardly distributed with UNIX operating systems, and may be combined within single line instructions to achieve efficient 'one-off' processing operations, or saved as executable shell script files for later use.

Within this setting, we have developed and refined C-coded tools for computing complexity and entropy from symbolic encodings of real-valued time series. The underlying string parsing operations embody typographical rather than numerical processing not easily handled within a mathematically oriented environment. Combining these with other standard UNIX tools enables us to compute the bulk of our results. Our use initial of Matlab was thus limited to peripheral computations and graphical display of the results. With time, this mode of working presented us with a bottleneck thus motivating the development of $\mathcal{S_P O_D}$.

$\mathcal{S_P O_D}$ is built around the OpenGL and GLUT libraries and operates as a stand alone *tool*, complementing our existing computational capabilities. It operates as an interpreter, translating static graphical prescriptions into 3D visualisation experiences. Its ability to accept large numbers of simple graphical objects gives access to visual effects that are geared to data 'scoping'. Interactive features, enable flexible animation and manipulation of the objects.

Our early focus was on improving visual-bandwidth over a narrow range of graphical functions. However, its capabilities have expanded considerably

R. Titchener M. (2006).
TOWARDS VISUAL-BANDWIDTH: GETTING CLOSE TO ONE'S EXPERIMENT DATA.
In *Proceedings of the First International Conference on Computer Graphics Theory and Applications*, pages 378-383
DOI: 10.5220/0001354003780383

beyond the initial requirements and is broadly useful in experimental research settings.

Separation of visualisation issues from the computational, created initially unforeseen opportunities at the data interface and invites creative use of the data structures both for visualisation as well as the subsequent data processing steps. Thus $\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$ has considerably changed the way we now undertake our research.

2 SLEEP-STATE RESEARCH

Determining patient sleep-state from recorded polygraph time-series data, usually requires a skilled professional, trained to recognise the salient features indicative of brain state. The process involves scanning visually through a night's polygraph recordings and manually scoring the events. The EEG (electroencephalogram) and EOG (electro-oculargram) channels are of prime interest (Sleigh et al.,) but anywhere from 16 channels of data are typically recorded from a patient in a diagnostic session. The recorded data (sampled time-series) include complex signal behaviors recorded from electrodes attached to the scalp. It is well understood that the EEG and EOG signals alone are highly indicative of brain sleep-state.

At, 50-100Hz sampling rates, roughly 50 megabytes of data is generated per patient per night. In a clinic, as many as six patients might be monitored simultaneously with the accumulated data amounting to gigabytes per month. While software applications with pattern recognition capabilities may be used to facilitate in manual scoring of the traces, the task remains essentially an operator intensive one.

Archived data is used as our principal resource for developing and testing new algorithms. In this respect data sets are constantly revisited, reprocessed and new results evaluated and compared against old results. Visual-bandwidth is key. The focus of our work is to develop automated algorithms sensitive to sleep state, to improve existing treatments for sleep disorders.

The brain may be viewed as complex non-linear dynamical system (Schuster and Just, 2005) that exhibits to varying extent, characteristics of deterministic chaos. The techniques we exploit to quantify system state draw from relatively recent results in non-linear dynamics (Steuer et al., 2001) relating entropy to Lyapunov exponents, and use a computable T-entropy(Titchener et al., 2005) thus as a measure of state. Entropy is a measure of chaos, of unpredictability or 'randomness'. By computing the information rate (T-entropy) from coarse grained encodings of sampled time-series data one may derive useful information about the corresponding sleep processes.

The EEG/EOG time series-data are saved as lists of space-delimited floating-point values for processing. The lists are recoded as binary strings through the application of a coarse grained bi-partition, or threshold. By means of 'scanning' the threshold across the signal dynamics, a series of encodings may be generated that together encapsulate the whole of the signal.

A sliding-window defines a current set of substrings from which T-entropy is compute. The values computed for each corresponding position of the window gives rise to a two dimensional array of values that may be visualised as a surface. The height of the surface corresponds to the information-rate conveyed by the time series samples averaged over a duration corresponding to the window width. For EEG and EOG signals these surfaces appear much like 'mountain ranges'. Lighting and surface orientation may be controlled to help identify salient sleep events (c.f. Figure 3).

The processed data from these channels is subsequently combined and displayed to advantage in $\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$, to extract more directly the sleep-state as a function of time. These techniques are not specific to this area of research, but may be easily adapted to areas ranging from seismology to biology, applied physics, medical diagnostics, as well as in engineering and industrial process control applications.

3 THE DATA INTERFACE

$\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$ operates as an interpreter, accepting preformated data either as standard input or to be read from a specified input file, prescribing surfaces, static or animated graphs and traces, and other positional markers. For readability, object data is stored as text arrays or lists of floating format numbers, reflecting the form of the experiment data itself. Viewing is a 2D projection of a working 3D volume, nominally defined by ($\pm$10, $\pm$8,$\pm$8) units.

Data headers associated with the data lists prescribe i) the display mode, ii) array size, and iii) the scaling, and iv) offsets required to place objects within the viewing space. The data lists are invariably remain in the raw unscaled form, useful for subsequent data processing steps. The scaling and offset calculations required in displaying the objects occurs as an overhead at the time objects are loaded into $\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$'s memory.

Illuminated scales (similar to an oscilloscope) are helpful in resolving orientation and visual scaling. The graphical object prescriptions include tags/labels that can be used to identify individual objects in, for example, an animated sequence. Optional qualifiers may be associated with individual objects and prescriptions, to preset viewing options and display modes. These may also be controlled interactively during viewing. Objects including surfaces, static and animated graphs, and so on, may selected individually, or in combination for viewing.

The focus of $\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$ has been on providing 'scoping' options, to allow one to explore data sets with ease, to extract or isolate events that can often be fleeting or transitory phenomena.

$\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$ is developed around the OpenGL (Kilgard, 1995) and GLUT (utility toolkit) (Kilgard, 1996) libraries. These largely address any portability issues. $\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$ essentially bridges the gap to take raw data into the required to exercise the features of OpenGL. $\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$'s interface gives the user the freedom to move around data sets in both space and time. Animation often plays an important role in discovering important or interesting effects. Much of its responsiveness derives from the graphics display hardware that is today ubiquitous across all computing platforms. Real-time animation of sequences of graphs, with rotation and rolling of viewing and light source positions enables one to quickly identify interesting aspects of one's data. Even changing the speed of animation can highlight what might be otherwise invisible dynamical effects. $\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$ can emulate the oscilloscope function but instead of being restricted to 2D, and two input channels, it provides support in effect for an unlimited number of channels and in a 3D environment.

The emphasis is on 'visual bandwidth' rather than 'gloss'. Whereas most graphics oriented applications dwell on the frills, providing attractive 2D depictions and 3D projections of charts, graphs, histograms, surfaces, replete with numbered / labeled axes, reference scales and grid, legend and/or other visual clutter, $\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$ follows the more simple display paradigm of Space Odyssey.

In a research setting one is generally less concerned with gloss, and more with exposing subtle features, trends, patterns, interesting artifacts, and so on. Turn-around-time from inputing one's data to visualising it and/or its processed derivatives takes precedence. 'Visual-bandwidth' is about giving *visual* access to data, rendering 2 & 3D graphical objects often in combination, allowing objects to be spun around, coloured and shaded, or otherwise animated, scanned, perused, to enable comparisons or reveal subtleties. In this respect, an interface ought not stand between the user and his or her data.

3.1 Object File Format

Orientation of the xyz axes in the viewing space follows a 'CNC machining' convention. Thus in the default position, the $x-$axis lies across the width of the screen, the $z-$axis corresponds to the vertical, and the $y-$axis points into the screen. Illuminated $z : x$ and $y : x$ scales divide the viewing space into quadrant volumes and invariably help with identifying the viewing orientation.

Simplicity is key to the graph prescriptions with the data presented simply as space delimited arrays of floating format numbers. Formatted headers prescribe how the associated data is to be rendered. The first line of an object prescription has the general form: object_specifier:[colour]:label. The object specifier comprises a single letter indicating the display type. As a rule, lower case specifiers are reserved for static objects, and upper case for animated or dynamic display objects.

- s: surface, rectangular mesh, displacements in z direction
- x: static 3D connected- or point-graph
- y: static 2D connected-or point-graph (parallel to yx plane)
- z: static 2D connected-or point-graph (parallel to zx plane)
- k: static 3D connected-or point-graph (parallel to zx plane)
- X: static or animated 3D connected-graph
- Y: dynamic 2D connected trace or sample point display (parallel to yx plane)
- Z: dynamic 2D connected trace or sample point display (parallel to zx plane)

The second character field in a header indicates graph colour (r: red, g: green, b: blue, y: yellow, c: cyan, m: magenta, w: white, <sp>: black). An extended colour set is accessed by using a further *object qualifier*. Such 'qualifiers' follow the object prescription which it amends or extends.

Each object is labeled. Distinguishing labels provide convenience both for identifying the individual graphs in animated sequences, but also when using a text editor to peruse or make quick changes to object specifiers. (A text editor also offers a convenient way to combine, i.e., cut and paste together new combinations of existing objects). In an extended sequence of animated graphs, label field may include a numbered index or other visual cues helpful when interpreting the sequence.

Animated displays are driven by two internal floating point parameters; a counting index g, that runs over the unit interval, $g \in [0.0, 1.0]$, and a *display range* index $h \in [0.00025, 1.0]$. Both may be controlled from mouse & keyboard actions, but may be preset by way of qualifier instructions. Within the present limits of the viewer, and its relatively simple underlying object format, there exists considerable flexibility for creative display of combined static and animated graphical data. These are partially illustrated in some of the figures that follow.

3.1.1 Surfaces

Surfaces are limited to regular 2D arrays: $z_{x,y} = s(x, y)$ defines the surface function s. This is sufficient for surface representations for a wide range of experiment data. A maximum of ten surface prescriptions may be loaded for viewing in a session and may be selected/deselected for viewing from key actions. Surfaces comprising up to 200,000 vertices are comfortably handled but more typically comprise 20–50,000 points for entropy surface descriptions arising in our sleep-analysis.

The surface object is indicated by the lower case specifier, `s`, followed by colon separator, and label text. Two integers follow to define the associated data array size. Six further parameters indicate the x-, y-, z- floating-point scale-factors and offsets, used to size and position the surface within the field of view. Then follows the surface data, an array of m lines each of n floating point values. Surface data is invariably kept in its raw form suitable for further processing if required. The format of the floating format list is not critical though will typically reflect the array dimensions. Data may be wrapped for improved readability, for example, in a text editor.

The surface display is modal. Key options chose between i) the wire geometry tessellated surface, ii) an opaque surface, with elements coloured as a function of the surface displacement, or iii) with surface smoothing (default display mode). The surface colour gradient, and brightness may be preset as part of the input file, but also changed interactively at the keyboard. The orientation of the object and light angle may also be specified as part of the input file. Again these are controllable from mouse and key combinations. The inclusion of single-line qualifying instructions in the object data file provide a convenient way to preset parameters so that a file can be opened to a preferred view. The current view settings may be output (with std output) to be saved with the object file to set the new default (opening) view.

3.1.2 Static Graphs

Static connected- or point-graphs may be prescribed in number of ways. For many situations it is sufficient to specify graphs "in the plane", for example, the yx plane or the zx plane. A `y` or `z` object specifier, indicates this accordingly. In either of these graphing modes, it is sufficient to simply list the data points to be graphed as a one dimensional array. As with the surface prescription, the array size parameters appear in the second line of the header. The array index implicitly defines the x-coordinates. The graph width is scaled to fit the viewing dimensions specified as part of the six x-,y-,z- *range* and *offset* parameters. Offsets are used to arrange visual separation between multiple graphs within the 3D display volume.

Static 3D 'trajectories', with the x co-ordinate representing the time scale for example and y and z values plotted as a function of array index x, use the `k` ('kurve') specifier. The 'kurve' option allows for a trajectory to be partitioned interactively and colour coded according to the segmented regions. The coloured trajectory may be output (std output) as a numerically scored graphical object prescription (The output of $\mathcal{S_P O_D}$ can in principle be piped to itself to create a new instance of the display). These rather specific features, used extensively in our work on sleep studies, have also been found to be more broadly applicable, in other engineering related analysis. Alternatively, to plot free-form 3D graphs or points series, $[x_i], [y_i], [z_i]$, the `x` specifier is used.

The aforementioned header format is adhered to across all of the display options. It includes the array dimensions, as well as the x-,y-,z- *range* and *offsets* respectively. Since static objects are scaled irrespective of the array size, to 'fill' the dimensions indicated by the 'scale' parameters, separate data series may be compared on the same time scale, irrespective of the numbers of points in the respective series. This is especially convenient when looking at multiple data series generated perhaps using unrelated sampling rates, but sharing a common time scale.

As mentioned $\mathcal{S_P O_D}$ currently accepts up to 16,000 static free-form connected graphs in any given session. There is no reason to see this as hard limit, but as this number is more than sufficient for our current requirements we have not felt the need to go further. It would be unusual to wish to display simultaneously this number of graphs anyway. More usually such a set of graphs would be 'played' as an animated sequence to enable convenient observation of dynamical events in the data sets. See Figure 1 as discussed below.

3.1.3 Dynamic Graphs and Animated Sequences

Dynamic "oscillograph" displays, animated displays of time-series data and graph sequences, are indicated with upper case object specifiers, `X`, `Y` and `Z`. As with the `y`, `z` specifiers, traces are implicitly graphed against a time scale (the x-axis), in the yx and zx planes, respectively. Data arrays may include upwards of a million data points. An internal counting index operates as a pointer into the data array, and a range-index defines a 'viewing window size' on the trace being displayed. The windowed portion of the trace is further scaled and offset for display purposes, in accordance with the header, range and offset parameters.

Traces may be displayed statically, or dynamically at rates considerably in excess of 10,000 points per second (very much faster than the eye can follow!).

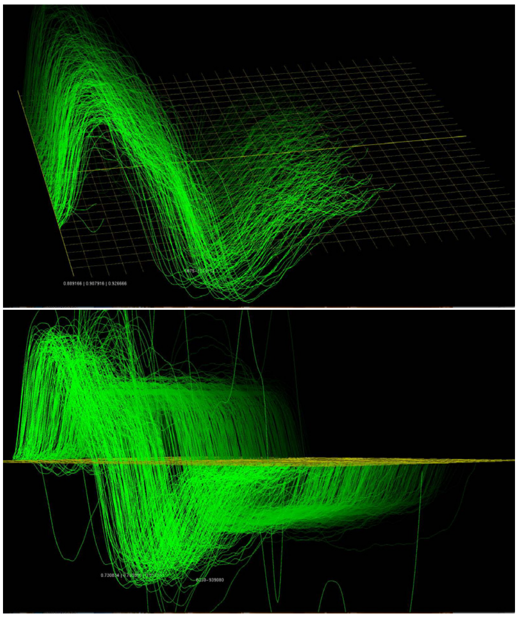

Figure 1: An animated sequence may be replayed with individual graphs separated in space to give the appearance of 'depth', or alternatively as a planar 'cascade'. The brightness of the graphs is modulated to add to the perception of depth. Rotation and lighting effects allow one to move around the objects dynamically to give advantageous views of changing effects.

Traces may also be manually scrolled across the viewing space using mouse and key actions.

A contrasting animation effect is possible using the `X` object specifier. As with the free from graphs (using the `x` specifier) objects comprise an array of $[x_i], [y_i], [z_i]$ co-ordinate values, scaled and offset in accordance with the header parameters. These objects default to being simultaneously visible unless a further instruction is included with the session. Alternatively such a sequence of graphs may be played in an animated sequence. An internal index counter determines the point in the graph sequence to be displayed, and an index-range parameter determines how many such graphs in the immediate vicinity of the current position pointer are displayed simultaneously.

Sequence replay is modal. Figure 1 shows a snapshot from a series of 7000 graphs, with two different views. In this viewing mode the graphs are placed in sequence along the y-axis to create a sense of depth that assists in visual interpretation of the data.

3.1.4 Mixing it all up

An important feature of $\mathcal{S}_\mathcal{P}\mathcal{O}_\mathcal{D}$ is its ability to simultaneously handle a mix of object types. Surfaces, static and animated graphs and curves may be combined to reveal relationships within a set of complex signals, and respective representations of these signals.

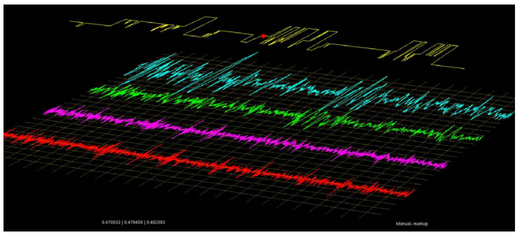

Figure 2: 'Snapshot' of animated polygraph data together with a static graph of the full night's sleep states (yellow, offset above the polygraph traces). Planar static graphs and animated traces, may be rotated and spun in the viewing space, give useful perspective on the data traces.

Figure 3 shows an example of graphical objects derived in relation to sleep analysis. The display corresponds to the complete night's data for one patient. The entropy surface computed from the EEG/C3 channel, encapsulates all of the sleep state activity for the night. The entropy surface cross-section at any given point in time (represented by x-axis) is a relatively precise, if not however convolved, measure of signal source statistics over the corresponding sample periods. The peak values for the entropy, defined by the surface ridge (and graphed in red directly above the ridge in the z-x plane), is graphed together with the maximum entropy values for the EOC/LOC (green). The latter is reflected in to the yx plane.

Plotting the maximum entropy values for C3 and LOC against each results in the 3D trajectory (multicoloured). This trajectory is partitioned using planar thresholds to divide the space into non-overlapping volumes, and colour-coded accordingly; blue : awake , red : REM sleep, green : S1, an NREM sleep state, yellow : S2,S3,S4 NREM sleep states. $\mathcal{S}_\mathcal{P}\mathcal{O}_\mathcal{D}$ then encodes these coloured coded segments into numerically labeled states, displayed as the graph in yellow below the surface.

4 CONCLUDING REMARKS

$\mathcal{S}_\mathcal{P}\mathcal{O}_\mathcal{D}$ offers a novel approach to data visualisation. As a interactive graphing and plotting tool, it accepts data either as standard input or from appropriately formated text files.

Graphical objects supported by the viewer include *surface arrays*, *static*- and *dynamic graphs* of various kinds, *positional markers* and combinations of these. As many as 16,000 objects currently may be simultaneously loaded for viewing. Surfaces are limited to 2D regular array of floating point values $z_{x,y} = s(x, y)$. Graphs in the zx or yx plane are prescribed as lists of floating point values $z_x = f(x)$, and $y_x = g(x)$, respectively. Trajectories may be prescribed either by a 2D array, listing points $y_x ==$

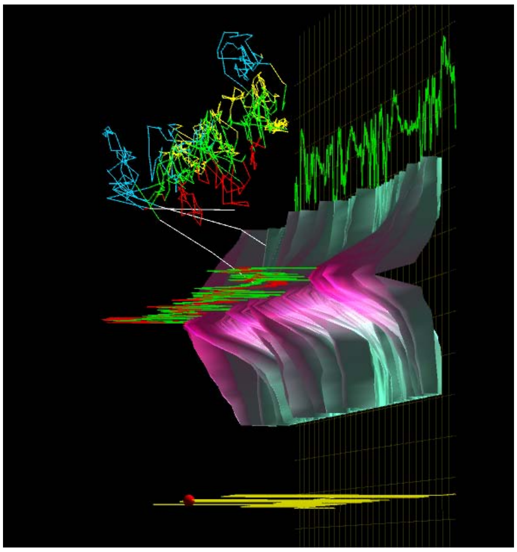

Figure 3: The above includes static graphs of the entropy of the EEG (red) and EOG (2 of green) channels. A 3D trajectory is generated (above left) by plotting EEG and EOG entropy series against one another. The trajectory is then further partitioned by way of 'invisible' intersecting planes and colour coded in respect of the volume associated sleep states. The largely obscured graph (yellow) is generated as a numerical encoding of the resultant states, and issues from $\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$ as *standard output*.

$f(x)$ and $z_x == g(x)$ or independently of x, by listing as a 3D array the coordinate points ($x = f(i); y = g(i); z = h(i)$, $i = 1, 2, 3 \ldots$). Graphs are displayed either as static or dynamic objects.

While many of the possible display modes have been outlined here, it is not easy to convey in words the full power of the dynamic and interactive aspects of the environment. The viewing experience, as with other 3D graphical environments, is hugely compelling in of itself, and $\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$ is currently finding application in other research areas, outside of the sleep studies.

Rather than being just another graphing tool, its convenience and power to flexibly present complex data, combinations of static and animated graphs, makes $\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$ equally useful in technical presentation settings. Though $\mathcal{S}_{\mathcal{P}}\mathcal{O}_{\mathcal{D}}$ has stabilised considerably since its inception, new features are being added at a steady pace, in response to new visualisation requirements, thus augmenting its already considerable versatility.

ACKNOWLEDGMENT

The author wishes to acknowledge that this research work has been carried out with support from Fisher and Paykel Healthcare (NZ) Ltd.

REFERENCES

Becker, R. A. and Chambers, J. M. (1981). *S: A Language and System for Data Analysis*. AT&T Bell Laboratories, Murray Hill, NJ, USA.

Char, B., Geddes, K., and Gonnet, G. (1983). The Maple symbolic computation system. 17(3–4):31–42.

Clarke, A. and Kubrick, S. (1968). *2001: A Space Odyssey*. MGM, Polaris productions.

Ihaka, R. and Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314.

Kilgard, M. (1995). OpenGL: Let there be light! *The X Journal: Computing Technology with the X Window System*, 4(3):12.

Kilgard, M. J. (1996). OpenGL: Free OpenGL software. *The X Journal: Computing Technology with the X Window System*, 5(5):74.

Kochan, S. and Wood, P. (1990). *UNIX Shell Programming*. pub-HAYDEN, revised edition.

Moler, C. B. (1980). MATLAB user's guide. Technical report, University of New Mexico. Dept. of Computer Science. Describes use of Classic Matlab, the prototype for the expanded professional Matlab from The MathWorks.

Schuster, H. and Just, W. (2005). *Deterministic Chaos, an introduction*. Wiley-VCH, Berlin, revised edition.

Sleigh, J. W., Steyn-Ross, D. A., Steyn-Ross, M. L., Grant, C., and Ludbrook, G. (2004). Cortical entropy changes with general anaesthesia: theory and experiment. *Physiological Measurement*, 25, pp 921–934.

Steuer, R., Ebeling, W. B., and Titchener, M. R. (2001). Partition based entropies. In *Stochastics and Dynamics*, pp 45–61. World scientific.

Titchener, M. R., Gulliver, A., Nicolescu, R., Speidel, U., and Staiger, L. (2005). Deterministic information theory. *Fundamenta Informaticae*, 64(1–4):443–461.