Data Representation for Scientific Visualization: An Introduction

Dan R. Lipşa[1], Robert S. Laramee[1], R. Daniel Bergeron[2], and Ted M. Sparr[2]

[1]Visual and Interactive Computing Group, Department of Computer Science, Swansea University, Swansea, UK
[2]Computer Science Department, University of New Hampshire, Durham, NH, USA

February 8, 2010

Abstract

Given the very rapid increase in today's computing power, scientists and researchers are able to collect ever larger repositories of data. A common way to explore and analyze this data is through visualization because of visualization's ability to provide rapid insight into large amounts of data. Scientific visualization, an important field of computer graphics, enables scientists to depict data from a real world phenomenon in ways that facilitate understanding of that phenomenon. Visualization often starts with data and a description of a problem. We describe common ways to represent scientific data, including a discussion of different grid types. These representations are determined by how the data is collected from experiments or produced by simulation and at the same time, these representations determine the kinds of visualization algorithms that can be applied on data. Considering that data represent continous phenomena, a process often needed in scientific visualization is the reconstruction of data at different positions in space than the sampling positions. The theory behind sampling and reconstruction of data, is presented. Wavelet transforms, a common way to deal with large data, are also introduced. Our goal is to provide a student or researcher new to the field of scientific visualization with a convenient, data-centric introduction into the field, and also provide an understanding of fundamental concepts required for further study.

1 Introduction

Most would agree that we live in an age characterized by an explosion in data and information. The very rapid increase in computing power often described

using Moore's law has enabled scientists, researchers and practitioners to collect, store and analyze unprecedented amounts of data. Tools that enable insights into these large quantities of data are of ever increasing importance.

Visualization is an important field of study that enables users to explore, analyze and present large amounts of data rapidly. Visualization as a branch of computer science is expanding and evolving quickly in response to the current data explosion. This is evident in the rapid increases in both commercial and open source visualization tools [14] as well as literature [4, 11]. Even Google has introduced a visualization API [3]. What we present is an introduction to the field of scientific visualization from a data-centric point of view. This paper provides the following contributions and benefits:

- We provide a concise introduction and a logical starting point for those interested or new to scientific visualization.
- We provide a description of scientific data, its characteristics and ways to model the data e.g. various grid representations
- Various filtering and interpolation methods that operate on and enhance the data are described.
- A brief introduction to both the theory and practice of data sampling and reconstruction is provided.
- Wavelets and wavelets transforms, an important mathematical tool for building multiresolution representations of data are introduced.

The resulting manuscript provides the reader with a concise and valuable introduction to scientific visualization starting with a data centric point of view. We aim to provide an educational tool for those new to the topic. The rest of this manuscript is organized as follows: the next section presents a brief description of scientific data characteristics and common processing. Those motivate the next sections: we present common types of scientific datasets and some of their storage methods, we briefly describe the theory behind the sampling and reconstruction of data points from a continuous function and we present the main concepts of wavelets and wavelets transforms.

2 Characteristics and common processing of scientific data

A scientific dataset consists of spatial data values either collected from the real world or produced by a computer simulation. Several different attributes (for instance the concentration of several chemical elements) can be acquired or produced for each point in space and these measurements can be repeated over a period of time. Scientific datasets have different characteristics based on the spatial arrangement of their data points: they differ in how they are stored and

in what algorithms can be used to process them. We present common types of scientific datasets and some of their storage methods.

Scientists explore scientific datasets in order to discover features of interest or to construct and study models of real world phenomena. Query operations are used to restrict the explored region of data, and visualization algorithms are used to display the data. A value for an attribute at a point in space can be thought of as being a sample from a function with values in every point in $\mathbb{R}^3$ where $\mathbb{R}$ is the set of real numbers. Often we need attribute values at other points in space than the sample data points contained in the dataset. The theory behind the sampling and reconstruction of data points from a function is summarized.

Often scientific datasets are very large, much larger than the main memory of a computer. Because of large disk-drive latency, visualization algorithms designed to process data from main memory cannot be directly applied to data stored on disk. One way to deal with large scientific datasets is to use a lower resolution version of the dataset for query, visualization and exploration. This can be done by using multiresolution (MR) and adaptive resolution (AR) representations of the dataset.

A multiresolution (MR) representation of the dataset consists of a hierarchy of versions of the original dataset, each having a different resolution. Typically a user starts by exploring the coarsest resolution version, which is much smaller than the original dataset. Higher resolution versions of the dataset are used either when the user zooms in locally to view a region of interest or when the user stops the exploration thus providing more time to render a higher resolution version of the dataset.

An adaptive resolution (AR) representation partitions a dataset into regions of different resolutions. Ideally, the resolution for each region is determined such that the error for that region is smaller than a user specified *error tolerance* for the dataset. A region of similar values will have a low resolution while a region with larger variations will have a high resolution such that the error for both regions will be smaller than the error tolerance specified for the dataset. The error for a region at the original resolution is zero. The error for a region at a lower resolution than the original is calculated by comparing the region at the original resolution with the region at the lower resolution.

Wavelets are a popular mathematical tool used for building a multiresolution representation of scientific data, image compression and in many other areas of computer graphics. We present the main concepts of wavelets and wavelets transforms.

3 Scientific Data Representation and Storage

In this section we present a model for scientific data. Based on the relationship between neighboring data points, scientific datasets can be represented as structured or unstructured grids. These are also described in this section.

3.1 A Data Model

Rhodes et al. [5] present a formal data model for scientific data which we briefly summarize in the next few paragraphs.

Scientific datasets represent a phenomenon defined over a continuous domain D. These datasets store values for a measured or simulated quantity that describes the phenomenon. Measurements are done and samples are recorded only for a finite number of points from the continuous domain D. If $\Delta \subset D$ is the set of sample points, the set of values stored in the dataset is given by a function $f_\Delta : \Delta \to \Omega$ where Ω is a subset of V, and V is the set of all possible values that the measured quantity can have. The error in each sample point is defined as a function $E_\Delta : \Delta \to E$ where E is the set of possible error values. Errors can appear because of the measurement devices or because of approximations and errors in the simulation. Formally, the dataset is represented [5] as a tuple:

$$R = \langle \Delta, \Omega, f_\Delta, E_\Delta \rangle \tag{1}$$

where Δ the set of all sample points, Ω the set of possible values for the measured quantity, f_Δ the function that gives the values measured at any sample point and E_Δ the function that gives the error at any sample point. The positions of all sample points is called the *geometry* of the dataset.

Data points in scientific datasets are normally part of a grid which is a tiling (tessellation) of the n-dimensional space, and the data model presented so far does not make provisions for that. Finding the neighbors of a data point and using tiles (cells) for processing the dataset is commonly used in scientific visualization so Rhodes et al. augment their model to contain a grid. The new model L which adds a grid component τ called the *topology* of the dataset to the previous model is presented in Equation (2).

$$L = \langle R, \tau \rangle \tag{2}$$

The scientific dataset model presented so far, which contains both the position in space of every sample point (the geometry of the dataset) and the grid information (the topology of the dataset) does not provide a description of the phenomenon measured over the whole continuous domain D over which the phenomenon occurs. To extend the measurements in the sample points to the whole domain D the authors use two *interpolation* functions. They use f_D to calculate the measured value in other points besides the sample points and they use E_D to calculate the error in other points besides the sample points. To calculate the value at any point in the domain D the interpolation function uses values in the sample points which are neighbors of that point. Both geometric and topological information may be used to do that. The new model is presented in Equation (3).

$$M = \langle L, f_D, E_D \rangle \tag{3}$$

When describing a grid of data points, there are two characteristics that are important: the structure of the grid which specifies the neighborhood information (topology) and the mapping between the structure and the position in space

of each point (geometry). There are two main types of grids: structured grid and unstructured (irregular) grid. In the interest of clarity, we present all the concepts in this section for two-dimensional data but they apply equally well to three-dimensional data.

3.2 Structured Grids

A structured grid of data points can be represented as a two-dimensional array, each position in the array storing all attribute values for a point. The coordinates in space of a data point can be calculated from its position in the array, together with some additional information that depends on the grid type. The neighbors of a data point can be retrieved implicitly from the position of the point in the array. These two characteristics lead to significant space saving when storing a structured grid dataset in memory or in a file.

Because it maps directly to a two-dimensional array, structured grid data is normally stored in a file using *linear storage*, by traversing its indexes (axes) in predefined order using nested loops [6]. To optimize disk access, a two-dimensional array is often stored in a file using *chunked storage* when data is split in tiles (chunks, cubes or bricks) of equal size, and each individual tile is stored contiguously in the file using linear storage. Tiles are stored in the file in linear fashion by traversing the axes of the volume in a certain order using nested loops. The order of traversing the axes and the size of the tiles is determined by the expected access pattern of the application.

With the same neighborhood information (structure) there can be different grid types based on the position in space of the data points. To specify the position in space of every point in the dataset we need to specify the position in space of the point at the origin of the array besides other information that depends on the type of structured grid that is used for the dataset. There are four different types of structured grids: regular, cartesian, rectilinear and curvilinear.

On a *regular grid* all data points are located on a tiling (tessellation) of the space with a rectangular tile. To specify the position in space of all points in the dataset, the additional information we need is the size (d_x, d_y) of the tile. Figure 1-a shows a two-dimensional regular grid.

A *cartesian grid* is a special case of a regular grid where all sides of the tile have the same length (i.e. $d_x = d_y$). To specify the position in space of all points in the dataset, the additional information we need is the length of a side of the square tile. Figure 1-b shows a two-dimensional cartesian grid.

Rectilinear grids have all data points on a tiling of the space with rectangular tiles of different size. These rectangles are specified by intersections of lines perpendicular on X and Y axes. All lines perpendicular to the X-axis are specified with an array of tile side lengths along X-axis and the same is true for all lines perpendicular to the Y-axis. So, to specify the position in space of all the points of the dataset, the additional information needed is two arrays of lengths, one for each axis X and Y. Figure 1-c shows a two-dimensional rectilinear grid.

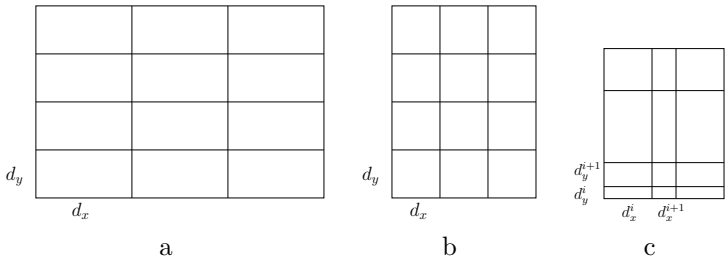

Table 1: a. Regular grid ($d_x \neq d_y$), b. cartesian grid ($d_x = d_y$), c. rectilinear grid ($d_x^i \neq d_x^{i+1}, d_y^i \neq d_y^{i+1}$).

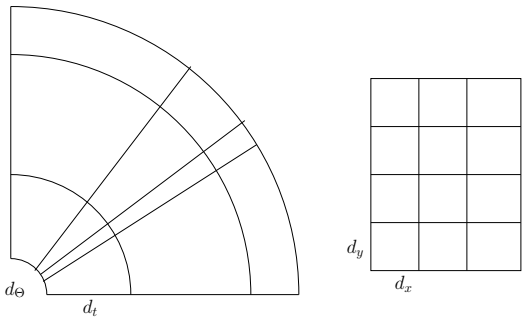

Figure 1: A curvilinear grid and a cartesian grid with the same topological structure ($d_t \rightarrow d_x, d_\Theta \rightarrow d_y$).

To understand a *curvilinear grid* we can think of a rectilinear grid with flexible edges, where the vertices of the grid are not in the original positions but are moved in space. It is possible for a curvilinear grid to have shared boundaries, for instance a 2D rectilinear grid that is wrapped around a cylinder. Sometimes, curvilinear grids specify the position in space of its points using a function of the point position in the structure array. It is also common for the position information to be stored explicitly in each point. Figure 1 shows a curvilinear grid (on the left) and a cartesian grid that has the same topological structure as the curvilinear grid (on the right).

3.3 Unstructured or Irregular Grids

An unstructured (irregular) grid data starts with data points at arbitrary positions in the two-dimensional space. To visualize data it is essential to have neighborhood (structure) information for the data points which is usually created with a procedure called triangulation. A triangulation of a set of points in the plane is obtained by adding the maximum set of edges that connect two points such that no new edge can be added without intersecting existing edges.

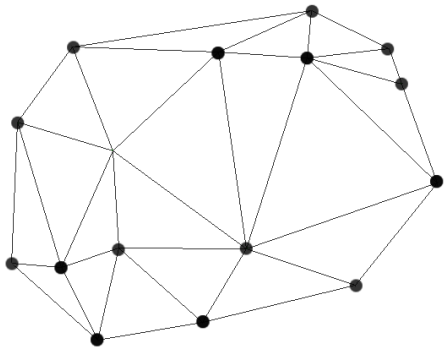

Figure 2: An unstructured (irregular) grid dataset

The data points together with the imposed structure determines a tiling of the two-dimensional space where each tile is a triangle. Both the position of points in space and neighborhood information needs to be stored together with data to enable efficient processing of irregular data.

Irregular volume data is usually stored in a format called *index cell set* (ICS) [7] as a list of vertices and a list of tetrahedral cells. For each vertex we store the x, y, z coordinates in space of that vertex and all data attributes. Each cell contains four indexes (references) to vertices in the vertex list.

4 Sampling and Reconstruction of Data

Often, visualization algorithms are used to process data obtained by sampling a function defined in all points in space. Ideally, the sampled function value, should allow us to reconstruct the original function without any loss, but often this is not possible. We would like to know how many samples we need from the original function such that reconstruction without loss is possible and how to reconstruct the signal such that we minimize the loss. The answers to these questions are provided by the field of signal processing [2]. In the rest of this section we present properties of functions of one variable (which can represent time or space) but the results we present can be extended to three or four dimensional functions that are used in scientific visualization.

4.1 Discrete Fourier Transform (DFT) and the Inverse DFT

An important tool in studying sampling and reconstruction of functions is Fourier analysis [1, 8, 12] which allows a function to be expressed as a (possibly infinite) sum of sinusoids of different frequency, amplitude and phase. In practice, the frequency, amplitude and phase for those sinusoids is calculated

using the Discrete Fourier Transform (DFT). Suppose we sample a function $x(t)$ at $t_0, t_1, ..., t_{N-1}$ to get N sample values $x_0, x_1, ..., x_{N-1}$. From the sample values we can define N sinusoids using the DFT formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i\omega_k t_n} \tag{4}$$

In Equation (4) $k = 0, \ldots, N-1$, x_n is the sample value for $t_n = nT$ the n^{th} sampling instant, T is the sampling interval, X_k is the amplitude and phase for $\omega_k = \frac{2\pi k}{N} f_s$ the k^{th} frequency sample and $f_s = 1/T$ is the sampling frequency. If x is a function in the temporal domain, its Fourier transform X is said to be the representation of the function x in the frequency domain. The Fourier transform of a signal is a complex function of the form $X_k = R_k + iI_k$ that encodes both the amplitude $|X_k| = \sqrt{R_k^2 + I_k^2}$ and the phase $\phi_k = \tan^{-1} \frac{I_k}{R_k}$ of the sinusoid with frequency ω_k.

The DFT formula is obtained by viewing the N signal samples as an n-dimensional vector in the vector space $\mathbb{C}^N$, where $\mathbb{C}$ is the set of complex numbers. In this space the set of vectors $s_1, s_2, \ldots, s_N$ is an orthogonal base where $s_k(n) = e^{i\omega_k t_n}$ is the complex sinusoid with frequency ω_k. The amplitude and phase X_k in the DFT formula is obtained by projecting vector x onto the k^{th} complex sinusoid s_k. All original signal samples x_n can be obtained by adding together all projections onto vectors s_k that form a base of $\mathbb{C}^n$. This transformation is called the inverse DFT:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{i\omega_k t_n} \tag{5}$$

The complex sinusoid s_k can be understood by using the Euler identity $e^{i\omega_k t_n} = \cos \omega_k t_n + i \sin \omega_k t_n$. Complex sinusoid s_k contains two sinusoids, one for the real and one for the imaginary axes of the complex plane. Note that both sin and cos can be thought of as sinusoids because $\cos \theta = \sin(\theta + \pi/2)$.

4.2 Graphical Representation of Functions in the Frequency Domain

Radian frequencies [8] $\omega_k = \frac{2\pi k}{N} f_s$ with $k = 0, 1, \ldots, N-1$ create N equally spaced frequency samples in the interval $[0, 2\pi f_s)$ with values:

$$0, 2\pi \frac{1}{N} f_s, 2\pi \frac{2}{N} f_s, \ldots, 2\pi \frac{N/2-1}{N} f_s, 2\pi \frac{N/2}{N} f_s, \ldots, 2\pi \frac{N-1}{N} f_s \tag{6}$$

You can add or subtract a multiple of $2\pi f_s$ from these radian frequencies without changing their meaning because $e^{i(\omega_k + 2\pi f_s)t_n} = e^{i\omega_k t_n} e^{2\pi f_s t_n}$. But $e^{2\pi f_s t_n} = e^{2\pi \frac{1}{T} T n} = e^{2\pi n} = 1$. An equivalent list of frequency samples obtained by subtracting $2\pi f_s$ from all frequencies greater or equal to $2\pi \frac{N/2}{N} f_s$ is:

$$0, 2\pi \frac{1}{N} f_s, \ldots, 2\pi \frac{N/2-1}{N} f_s, \underbrace{2\pi \frac{-N/2}{N} f_s, \ldots, 2\pi \frac{-1}{N} f_s}_{\text{negative frequencies}} \tag{7}$$

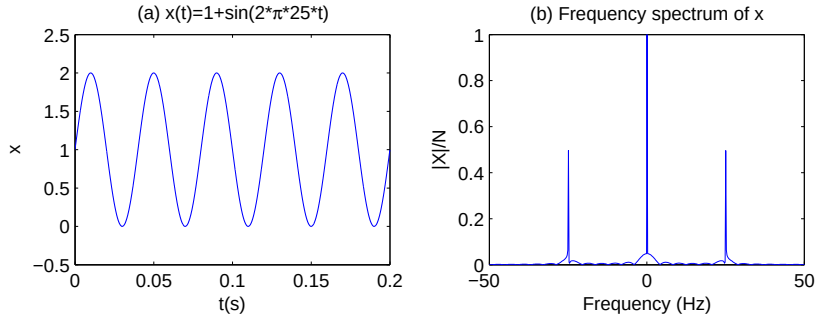

Figure 3: Frequency spectrum of a sin function: (a) $x(t) = 1 + \sin 2\pi 25t$ and (b) frequency spectrum of x, where $X = \mathrm{DFT}(x)$ and N is the number of samples.

The list of frequencies in Equation (7) can be reordered as:

$$\underbrace{-\pi fs, -\pi(1-\frac{2}{N})fs, \ldots, -\pi\frac{2}{N}fs}_{\text{negative frequencies}}, 0, \pi\frac{2}{N}, \ldots, \pi(1-\frac{2}{N})fs$$

The representation of a signal in frequency domain is usually shown with frequencies in the interval $[-\pi f_s, \pi f_s)$, where a negative frequency can be translated to a frequency in the interval $[\pi f_s, 2\pi f_s)$ as shown above. When the Fourier transform of a function x is displayed graphically, the phase ϕ_k is usually ignored and the amplitude $|X_k|$ is showed as a function of frequency ω_k. Figure 3 shows the graphs for $x(t) = 1 + \sin 2\pi 25t$ and for the DFT(x). Note that the DFT graph shows a zero frequency (DC - direct current) component which is equal to 1 and two complex sinusoids with frequency equal to -25 and 25 Hz and amplitude 0.5. The two complex sinusoids combine to yield a real sinusoid with frequency of 25 Hz and amplitude of 1. Note that for real signals x we have that $|X(\omega_k)| = |X(-\omega_k)|$ for all frequencies ω_k so the graphical representation of X_k is symmetric about axis Y. The amplitude of the real sinusoid is twice the amplitude of $|X(\omega_k)|$. This can be explained by using the Euler identity to express a real sinusoid as a sum of two complex sinusoids with the same absolute frequency and half the its amplitude:

$$\cos\theta = (e^{i\theta} + e^{-i\theta})/2 \quad \text{and} \quad \sin\theta = (e^{i\theta} - e^{-i\theta})/(2i)$$

4.3 The Nyquist Rate

It has been shown [8] that a signal can be reconstructed completely from its samples if the signal was sampled at a frequency that is greater than $2 * f_h$, where f_h is the highest frequency component in the signal. That minimum sampling frequency is called the *Nyquist rate*. Unfortunately, many signals in the real world have infinite frequency spectrum which means that those signals cannot

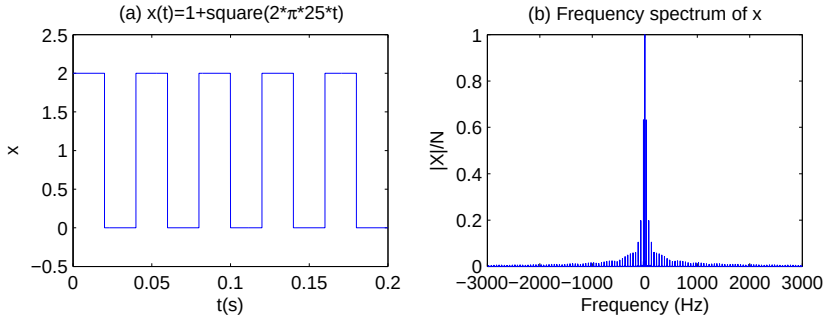

Figure 4: A square function and its frequency spectrum. (a) $x(t) = 1 +$ square $(2\pi 25t)$ and (b) frequency spectrum of x, where $X = \mathrm{DFT}(x)$ and N is the number of samples.

be reconstructed from a set of samples, regardless of the sampling frequency. For instance, a square wave has infinite frequency spectrum because its value changes discontinuously as showed in Figure 4.

4.4 Low Pass Filters and the Convolution Operation

An operation that is often useful, called low pass filter, removes all components with frequencies larger than a certain frequency ω from a signal. This operation makes it possible to sample and reconstruct the remaining signal without loss because its frequency spectrum is finite. A low pass filter in the frequency domain is realized by multiplying the signal in frequency domain with a pulse function

$$p(k) = \begin{cases} 1 & \text{if } |\omega_k| < \omega \\ 0 & \text{otherwise} \end{cases}, \text{where } k = 0..N-1$$

where N is the number of samples.

An image can be thought of as a function f of two variables where the color of the pixel at position (x, y) is given by $f(x, y)$. A low pass filter in the frequency domain on function f causes blurring in the associated image as this filter removes sharp transitions in a signal.

Figure 5 shows the signal $x(t) = 1 +$ square $(2\pi 25t)$ limited at 500 Hz in the frequency domain. To be able to reconstruct the remaining signal without loss we have to sample the signal at a sampling rate above the Nyquist rate, which is 1000 Hz. If the sampling rate for a signal is below the Nyquist rate higher frequency components in the signal can masquerade as lower frequency components, a phenomenon called *aliasing*. Figure 6 shows a sinusoid sampled at below twice its frequency. The samples obtained look like a sinusoid with lower frequency than the original.

To apply a low pass filter to a signal x we need to multiply X, the representation of the signal in the frequency domain, with a pulse function. Signal

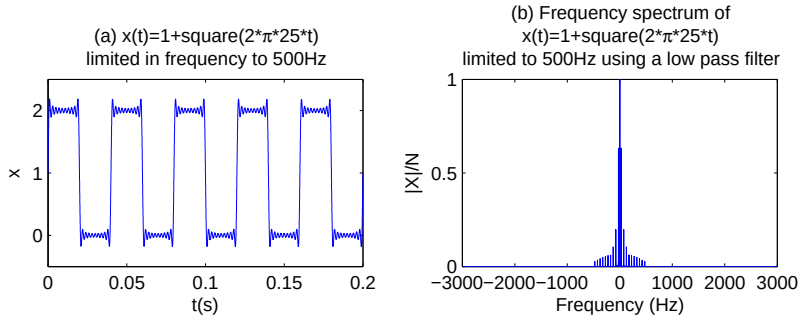

Figure 5: A square function ($x(t) = 1 + \text{square}\,(2\pi 25t)$) limited in frequency, (to 500 Hz) represented in time and frequency domains. Plot (a) show the time domain and plot (b) shows the frequency domain.

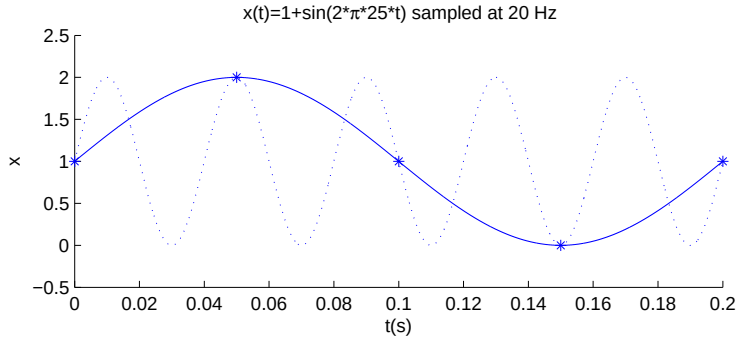

Figure 6: Aliasing: a signal sampled at frequency under Nyquist rate, can look like a lower frequency signal. In the figure $x(t) = 1 + \sin 2\pi 25t$ (interrupted line) is sampled at 20 Hz while its Nyquist rate is 50 Hz. Samples look as if they are part of a sinusoid with frequency 5 Hz ($1 + \sin 2\pi 5t$ drawn with a continuous line)

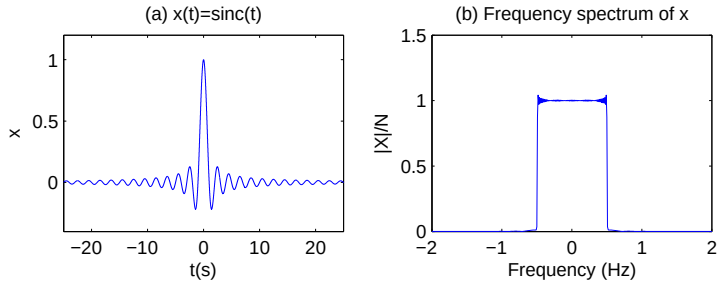

Figure 7: A low pass filter in the frequency domain, and its representation in the time domain, the sinc function. (a) $x(t) = \text{sinc}(t)$ and (b) the pulse function, sinc representation in the frequency domain. To obtain a perfect pulse function in the frequency domain, x has to be transformed using the Discrete Fourier Transform over $(-\infty, \infty)$.

processing theory shows that multiplication in the frequency domain of X with a function Y corresponds to $x * y$ in the time domain, where $*$ is an operation called convolution and x and y are the Inverse Fourier Transform of signals X and Y respectively. For discrete signals, the convolution between signal x of length n and signal y of length m is a signal w of length $m + n - 1$. Algebraically, the convolution is the same operation as multiplying the polynomials whose coefficients are elements of x and y.

$$w(k) = \sum_j x(j)y(k-j) \quad \text{where } k = 0, \ldots, m+n-2$$

and the sum is over all the values which lead to legal subscripts for x and y.

4.5 Reconstruction Filters

It has been shown [13] that the Inverse Fourier Transform of a pulse function is a function called sinc where $\text{sinc}(t) = \frac{\sin \pi t}{\pi t}$. That means that to apply a low pass filter to a function in the frequency domain we need to convolve the function with a sinc function in the temporal domain. Figure 7 shows a sinc function together with its frequency spectrum.

Unfortunately, sinc has infinite domain, so in practice, the sinc function is approximated using a pulse, a triangle or a Gaussian function. Figure 8 shows a pulse, triangle and Gaussian functions in the temporal domain and in the frequency domain.

Suppose that a signal has its frequency spectrum limited so that a sample from the signal can be theoretically used to reconstruct the original signal without loss. This reconstruction can be done if the signal is sampled at a frequency twice as large as its largest frequency component. It can be shown that the frequency spectrum of the sample is obtained by duplicating the frequency

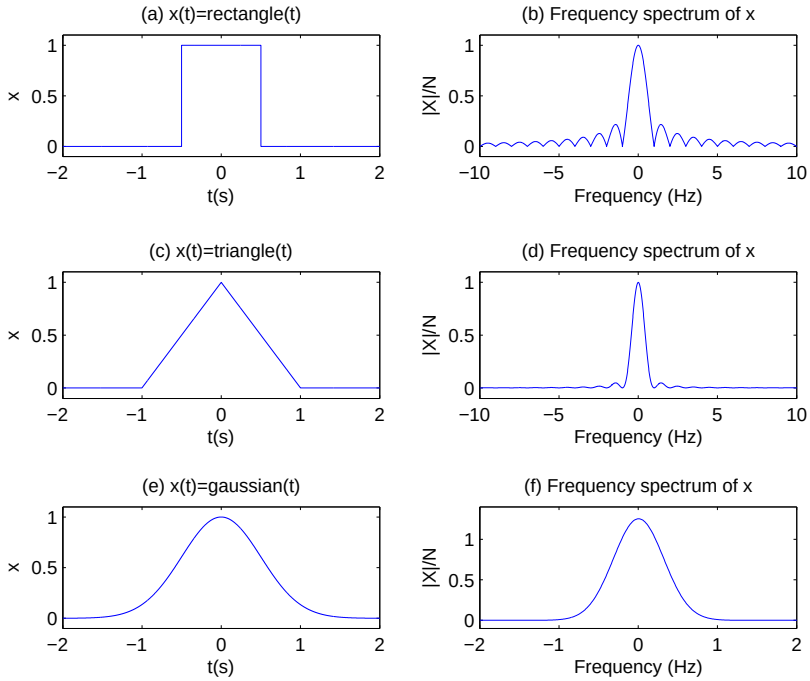

Figure 8: Approximations of the low pass filter (pulse function) in the frequency domain and their representations in the time domain: rectangle, triangle and Gaussian functions. The time domain is represented in (a), (c) and (e) and the frequency domain is represented in (b), (d) and (f).

spectrum of the original signal at equally spaced frequencies along the entire frequency spectrum. To reconstruct the original signal, the sample has to be multiplied with a low pass filter in the frequency domain or convolved with a sinc function in the temporal domain. We note that sinc has infinite domain so, in practice, the signal will be convolved with one of its approximations which has finite domain. That means that, in practice, the original signal cannot be accurately reconstructed even if sampling was done at above Nyquist rate. Differences between the original signal (frequency spectrum limited) and the reconstructed signal are called *artifacts*.

5 Wavelets Transforms

Wavelets [9, 10] are a mathematical tool for decomposing data into approximation and detail constituents. This decomposition can be applied recursively which yields a hierarchy of approximations to the original data from the original high resolution to a desired low resolution. Each of these approximations, together with detail data can be used to completely recover the original data. There are different kinds of wavelets which determine different ways of decomposing a dataset and can be applied to different kind of data such as an image, a curve, a surface or a volume. Wavelets are used in many areas of computer graphics such as image compression, level-of-detail control for editing, rendering surfaces and global illumination [10].

The computation that transforms data into approximation and detail components is called an *analysis filter* or a *decomposition filter* and the computation that recovers the original data from approximation and detail data is called a *synthesis filter* or a *reconstruction filter*.

As an example we apply an *analysis filter* to a unidimensional dataset to decompose it into an approximation part and a detail part. We are going to use the *Haar analysis filter*, a simple and widely used filter. Suppose that we start with a dataset with four values $[5, 3, 7, 9]$. To obtain the approximation part we average every two values in the list to get $[\frac{5+3}{2}, \frac{7+9}{2}]$ which is $[4, 8]$. To obtain the detail part we subtract every two values in the list and then we divide the result by two to get $[\frac{5-3}{2}, \frac{7-9}{2}]$ which is $[1, -1]$. From a dataset with four values we obtain an approximation dataset that has two values and a detail dataset that has two values. We can apply our filter again on the approximation dataset with two values to get an approximation value and a detail value. The entire process is summarized in Table 2:

Resolution	Approximation	Detail
4	$[5, 3, 7, 9]$	
2	$[4, 8]$	$[1, -1]$
1	$[6]$	$[-2]$

Table 2: Haar analysis filter

The one approximation value for the dataset followed by detail values in or-

der of increasing resolution is called the *Haar wavelet transform* of the dataset. For our example the wavelet transform is $[6, -2, 1, -1]$. Note that using the wavelet transform, we can reconstruct any level of resolution including the original data.

To decompose a 2D dataset (for example an image), the Haar decomposition filter is generalized in two dimensions. The 1D filter is run on all rows of the dataset and then the 1D filter is run on all columns of the horizontal decomposition resulted in the previous step. This yields the average data stored in the upper left quadrant of the 2D dataset and detail data stored in the rest of the quadrants. The process is continued recursively, on the average data. (the next data that is split is the upper left quadrant). The Haar decomposition filter is generalized for 3D data in a similar fashion.

An alternate view of a sequence of values is to view it as a piece-wise constant function defined on the interval $[0, 1)$ with values in the set of real numbers. For example, a sequence with two values can be seen as a function which has two constant values over intervals $[0, 1/2)$ and $[1/2, 1)$. The set of all functions with two values is denoted with V^1 (2^1 is the number of values that a function in the set can take). A sequence with four values can be seen as a function which has four constant values over intervals $[0, 1/4)$, $[1/4, 1/2)$, $[1/2, 3/4)$, $[3/4, 1)$ and the set of all functions with four values is denoted with V^2 (2^2 is the number of values that a function in the set can take). A sequence with 2^j values can be seen as a function which has constant values over equal intervals of size $1/2^j$ and the set of all functions with 2^j values is denoted with V^j.

Each set V^j can be thought of as a vector space, where each function with 2^j values is a vector in this space. A basis for this vector space is the following set of scaled and translated "box" functions:

$$\phi_i^j = \phi(2^j x - i). \qquad i = 0, \ldots, 2^j - 1.$$

where the box function ϕ is defined by:

$$\phi(x) = \begin{cases} 1 & \text{for } 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases}$$

For example, the basis for V^2 is showed in Figure 9. The function used in our previous example $[5, 3, 7, 9]$, can be expressed in terms of V^2 basis as: $f = 5\phi_0^2 + 3\phi_1^2 + 7\phi_2^2 + 9\phi_3^2$.

The set V^j of all functions with 2^j values is included in the set V^{j+1} of functions with 2^{j+1} for all values j. So we have:

$$V^0 \subset V^1 \subset \ldots \subset V^j \subset V^{j+1} \qquad \text{for all } j \geq 0$$

We define the inner product between two functions f and g in the vectors space V^j as the integral of the product of the two functions: $\langle f, g \rangle = \int_0^1 f(x)g(x)\mathrm{d}x$. This can be used to define vectors in V^j that are orthogonal to each other, as the vectors for which the inner product is zero. We can now define a set, denoted with W^j, which is the orthogonal complement of V^j in V^{j+1}. This means that

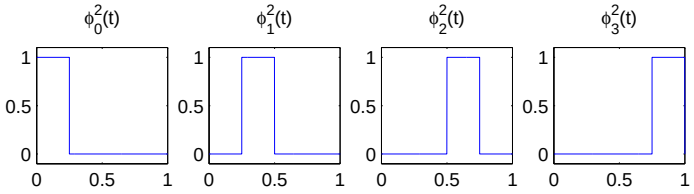

Figure 9: Any sequence of four values can be seen as a function that has four constant values over intervals $[0, 1/4), [1/4, 1/2), [1/2, 3/4)$ and $[3/4, 1)$. Each of these functions, in turn, can be seen as a vector in a vector space denoted with V^2. This figure shows a basis for V^2 formed by scaled and translated box functions

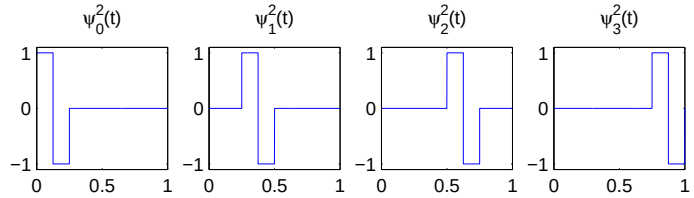

Figure 10: Haar wavelets for W^2 or basis for vector space W^2. W^2 is the complement of V^2 (vector space of functions with 4 values) in V^3 (vector space of functions with 8 values) so $V^2 \cup W^2 = V^3$

$V^j \cup W^j = V^{j+1}$ and every vector in W^j in orthogonal to every vector in V^j. A basis for W^j, known as the Haar wavelets are given by:

$$\psi_i^j = \psi(2^j x - i), \qquad i = 0, \ldots, 2^j - 1$$

where

$$\psi(x) = \begin{cases} 1 & \text{for } 0 \leq x < 1/2 \\ -1 & \text{for } 1/2 \leq x < 1 \\ 0 & \text{otherwise} \end{cases}$$

For example the Haar wavelets for W^2 are showed in Figure 10. The decomposition presented in Table 2 can be thought as a change of basis in vector space V^2. We start with the box functions basis: $f = 5\phi_0^2 + 3\phi_1^2 + 7\phi_2^2 + 9\phi_3^2$. We express the vector in V^2 in terms of the bases for V^1 and W^1: $f = 4\phi_0^1 + 8\phi_1^1 + 1\psi_0^1 + (-1)\psi_1^1$. Then, we express the vector in V^1 ($[4, 8]$) in terms of the bases for V^0 and W^0: $f = 6\phi_0^0 + (-2)\psi_0^0 + 1\psi_0^1 + (-1)\psi_1^1$. This is the Haar wavelet transform of the original function.

The Haar wavelet transform can be extended in two and three dimensions. For example in two dimensions we can apply the Haar filter on rows and then apply the Haar filter on columns to get an approximation that is one fourth of the size of the original data plus detail coefficients that are three fourths of the

size of the original data. The process continues until we get a value which is an approximation of the entire dataset and detail values.

6 Conclusions

We have provided a concise introduction into the field of scientific visualization from a data-centric point of view. We have presented scientific data characteristics and common processing and those motivated the next topics of our manuscript. We have described a formal data model and common types of data representation and storage for scientific data. We have briefly presented the theory behind sampling and reconstruction of data. Wavelets and wavelets transforms, a common way to build a multiresolution hierarchy for dealing with large data was introduced. We hope to have provided a student or researcher new to the field of scientific visualization with an understanding of fundamental concepts used in the field, and a good starting point for further study.

References

[1] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*, chapter 14, pages 617–646. Addison-Wesley, New York, NY, USA, 1996.

[2] R.C. Gonzalez and R.E. Woods. *Digital Image Processing, Third Edition*, chapter 4. Pearson Prentice Hall, Upper Saddle River, NJ, USA, 2008.

[3] Google. Google Visualization API, 2008. online document, accessed Jul. 2009, published Mar. 2008, `http://code.google.com/apis/visualization`.

[4] C. Hansen and Editors C. Johnson. *The Visualization Handbook*. Elsevier Academic Press, Boston, MA, USA, 2005.

[5] Philip J. Rhodes, R. Daniel Bergeron, and Ted M. Sparr. A Data Model for Distributed Multiresolution Multisource Scientific Data. In *Hierarchical and Geometrical Methods in Scientific Visualization*, Tahoe City, CA, USA, 2002. Springer-Verlag.

[6] Sunita Sarawagi and Michael Stonebraker. Efficient Organizations of Large Multidimensional Arrays. In *Proc. of the Tenth International Conference on Data Engineering*, pages 328–336, Washington, DC, USA, 1994. IEEE Computer Society.

[7] C. Silva, Y. Chiang, J. El-Sana, and P. Lindstrom. Out-of-Core Algorithms for Scientific Visualization and Computer Graphics, Course Notes for Tutorial 4. In *IEEE Visualization*, Boston, MA, USA, 2002. IEEE Computer Society Washington, DC, USA.

[8] Julius O. Smith. *Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications, Second Edition*. W3K Publishing, Standford, CA, USA, 2007. online book, accessed Feb. 12 2008, `http://ccrma.stanford.edu/~jos/mdft/`.

[9] EJ Stollnitz, AD DeRose, and DH Salesin. Wavelets for computer graphics: a primer. *IEEE Computer Graphics and Applications*, 15(3):76–84, 1995.

[10] E.J. Stollnitz, T.D. DeRose, and D.H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco, CA, USA, 1996.

[11] A. C. Telea. *Data Visualization, Principles and Practice*. A. K. Peters, Ltd., Wellesley, Massachusetts, USA, 2008.

[12] Larry N. Thibos. *Fourier Analysis for Beginners*. Visual Sciences Group, Bloomington, IN, USA, 2003. online book, accessed Sept. 05 2008, `http://research.opt.indiana.edu/Library/FourierBook/toc.html`.

[13] Larry N. Thibos. *Fourier Analysis for Beginners*, chapter 11. Volume 1 of *1* [12], 2003. online book, accessed Sept. 05 2008, `http://research.opt.indiana.edu/Library/FourierBook/toc.html`.

[14] F.B̃. Viegas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon. Many Eyes: A Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics,*, 13(6):1121–1128, 2007.