Chapter 20

INFORMATION DISPLAY

Peter Eades,[1,2] *Seokhee Hong,*[1,2] *Keith Nesbitt,*[3] *and Masahiro Takatsuka*[1,2]
[1]University of Sydney,
[2]National ICT Australia,
[3]Charles Sturt University, Australia

Recent increases in the size of available datasets have created a strong demand for new ways of displaying information. This chapter describes some recent research into information display.

Visualization is the process of generating a picture of a dataset. The data, which may be numerical, ordinal or nominal, is mapped onto visual variables so that they can be visually inspected. The visual variables determine the shapes and appearances of pictorial icons representing the data. These pictures are then placed on a display screen. The main challenge is to convey as much information as possible when these icons are displayed.

In many cases, the data are modeled as a graph, and the visualization process is called *Graph Drawing*. Section 1 of this chapter describes new methods for drawing graphs, aimed at coping with very large data sets.

The underlying display technology for visualization is undergoing rapid changes. In Section 2 we describe some of these new display technologies. These changes take information display beyond the visual; new methods for showing information using the nonvisual senses are described in Section 3.

1 NEW METHODS FOR DRAWING VERY LARGE GRAPHS

Much of the information and data in real-world applications consists of entities and the relationships between the entities, and thus can be modeled mathematically as *graphs*. For example, traditional entity-relationship diagrams and UML diagrams in software engineering can be modeled as graphs. Biological data such as phylogenies can be modeled as trees, PPI (Protein–Protein Interaction)

networks can be modeled as graphs, and metabolic (or biochemical) pathways can be modeled as directed graphs. Network data, such as webgraphs, and social network data can be modeled as undirected and directed graphs.

Graph drawing aims to construct good *drawings* (that is, visualizations, or *layouts*) of graphs in two or three dimensions. As the examples given above indicate, Graph drawing systems can be used in many applications such as software visualization, bioinformatics visualization, VLSI design, network data visualization, and social network visualization.

The main challenge in graph drawing is to design efficient algorithms and methods for computing good geometric representations of graphs automatically. There is a great deal of literature in graph drawing, and this research area has been growing for the last decade. Several books are available; see [67, 77, 79, 85]. There is an annual symposium on graph drawing that brings together mathematicians, computer theoreticians, and practitioners.

Further, many fundamental algorithms in graph drawing have been successfully developed and implemented by researchers and software developers. As a result, graph drawing systems and commercial software products are available. These include *GraphViz* from AT&T, *GDToolkit*, *AGD*, *Graphlet*, *TomSawyer Software*, and *ILOG*. For details, see the recent book on graph drawing software [77]. These products are successfully used for software engineering, network data analysis, and visual analysis of bioinformatics data.

The methods and algorithms for graph drawing can be roughly partitioned on the *types* of graphs, *edge representations* and *aesthetic criteria*.

The main types of graphs are trees (rooted trees or free trees), planar graphs, undirected graphs, and directed graphs.

Edges may be represented as straight-line segments, polylines, or orthogonal polylines (of horizontal and vertical line segments). Directed edges normally have arrowheads.

Aesthetic criteria are objective functions for optimization, defining *"good"* visualization of graphs. In general, they measure the readability of a drawing. Sometimes they relate to a specific application domain. For example, when drawing organization charts for a work group, it is important for the boss to be at the top of the page. On the other hand, there are a number of criteria that are independent of the application domain; the most important are the following:

- Minimizing the number of edge crossings
- Minimizing the drawing area (thus maximizing the resolution for a fixed-size screen)
- Maximizing the number of symmetries
- Minimizing the number of bends (in a polyline drawing)
- Minimizing the total edge length
- Uniform edge length
- Good aspect ratio, that is, balancing the width and the height
- Maximizing angular resolution, that is, ensuring that two edges adjacent to a vertex are drawn with enough angular difference

Unfortunately, achieving these aesthetic criteria is very difficult. For example, the problem of constructing a drawing of a general graph with a minimum number of edge crossings is NP-hard. This also holds for most of the other aesthetic criteria.

However, many efficient polynomial time algorithms have been developed for restricted classes of graphs, such as trees and planar graphs.

Also, many practical heuristic approaches have been successfully developed for general graphs. The best known of these heuristic approaches are the *spring algorithms* (or *force directed methods*) for undirected graphs and the *Sugiyama* (or *layered drawing*) *method* for directed graphs. For an overview of each method, see [67]. Both spring methods and Sugiyama methods are popular and widely used in many applications. In the remainder of this section, we describe very recent force-directed methods.

In general, spring algorithms use a physical analogy for graph drawing. For example, the edges of graphs can be replaced by springs to define attractive forces between the two vertices, and repulsive forces can be defined for each pair of vertices to guarantee that they are not drawn too close to each other. Then the system tries to achieve the minimum energy state (or equilibrium state), where the sum of all forces acting on each vertex becomes zero.

Due to the simplicity of the method and the reasonable quality of the drawing, many variations on the spring algorithm have been developed over the last two decades. Early examples include the *spring embedder* by Eades [68], forces using graph theoretic distance by Kamada and Kawai [78], and a *magnetic spring algorithm* by Sugiyama and Misue [86]. These methods differ slightly in terms of the force model and the method to reach equilibrium. For some comparison, see [67].

However, these early methods exhibit relatively high running time, iteratively computing $O(|V|^2)$ forces, where $|V|$ represents the number of vertices in the graph. This limits the size of graphs that such methods can handle in practice: in 1984 Eades reported a limit of about 50 vertices [68], and with 2004 technology these methods can handle a few hundred vertices. For larger graphs, either the quality is poor or the run-time is unacceptable for real-time visualization.

The size of data in practical applications has also grown in the early twenty-first century. For instance, the size of webgraphs is typically measured in the millions. Hence, the scalability of spring algorithms has been a challenging problem.

Recently, many new methods have been successfully developed to solve the problem of drawing very large graphs. Here we briefly describe the main ideas and results. In particular, we review four different approaches:

- Multilevel approach for the force-directed method
- Force-directed method using geometric clustering
- High-dimensional approach
- Spectral method

1.1 Multilevel approach

Walshaw presents a heuristic that uses a multilevel technique combined with a force-directed method [87, 88]. The main idea is to apply a kind of combinatorial clustering (or graph partitioning) method to gradually reduce the size of the

graph. If the size becomes small enough, then we can draw the small graph to produce an initial drawing. Then we gradually refine the drawing using a simple interpolation technique and a force-directed method to obtain a drawing of the original graph. More specifically, the multilevel process works as follows.

The first step is to group pairs of vertices to form *clusters*, using a fast heuristic for matching. Then the clusters define a new *coarsened* graph. This step is repeated until the size of the graph falls below some threshold. The second step is to draw the coarsened graph with a random initial drawing. The final step is to successively refine the drawing of the coarsened graph to get a drawing of the original graph, using a simple interpolation technique together with a modified version of the Fruchterman and Reingold force directed method [70].

It is claimed that the running time at each level is approximately $O(|V| + |E|)$ for sparse graphs. However, the total running time may depend on the number of levels of the multilevel process.

The method can compute both two- and three-dimensional drawings, and experimental results have been demonstrated with a number of examples from a few hundred vertices up to 225,000 vertices. The method works very fast, in particular, for 2D drawings of sparse graphs; for example, it takes around 30 seconds for 10,000 vertices. It may takes 10 minutes for the largest graph. For details of experimental results, see [87, 88].

Figure 20.1 shows two examples produced by the method [87]. Figure 20.1a is a drawing of a graph with $|V| = 4970$ and $|E| = 7400$. It takes about 14 seconds. Figure 20.1b shows a drawing of the graph *sierpinski10*, which has $|V| = 88575$ and $|E| = 177147$. It takes about 217 seconds.

Similar ideas were independently used by a number of authors, including Hadany and Harel [74], Harel and Koren [75, 76], and Gajer, Goodrich, and Koburov [71].

Harel and Koren [75, 76] used a *multiscale* technique, with a version of the algorithm of Kamada and Kawai [78]. Their method computes a sequence of improved approximations of the final drawing. Each approximation allows vertices to deviate from their final place by an extent limited by a decreasing constant. As a result, the drawing can be computed using increasingly coarse representations of the graph, where closely drawn vertices are collapsed into a

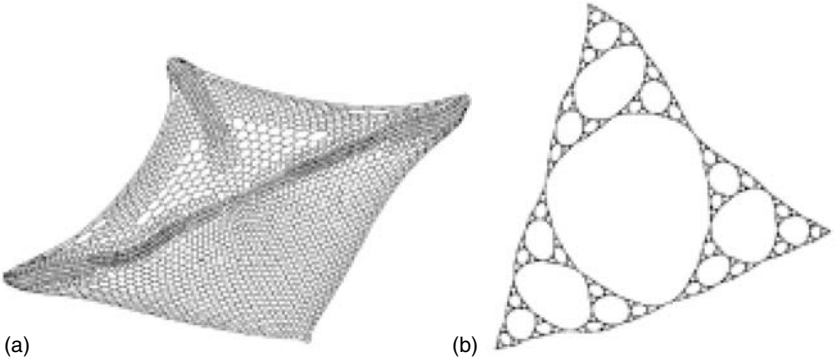

Figure 20.1. (a) Drawing of graph with 4970 vertices; (b) drawing of *sierpinski10* [87].

single vertex. Each drawing in the sequence is generated quickly, performing a local beautification step on the previously generated drawing. This method can handle up to a few thousand vertices. For details, see [75, 76].

Gajer, Goodrich, and Koburov also used a similar *multidimensional* technique for drawing large graphs [71]. The algorithm is implemented as a system called *GRIP* [72]. For details, see [71, 72].

1.2 Graph drawing using geometric clustering

Another force-directed approach using *geometric* clustering is presented by Quigley and Eades [83]. The algorithm is an extension of the Barnes–Hut hierarchical space decomposition method [63] to forced directed graph drawing. The main idea is to use a decomposition tree to approximate the force computation between each pair of vertices. Roughly speaking, these pairs approximate forces between vertices based on geometric clustering, defined by the decomposition tree. More specifically, the forces between *close* vertices are computed by the standard direct repulsion between two vertices, whereas the forces between *distant* vertices are computed using a geometric clustering induced by the decomposition tree.

Quigley and Eade's method uses a recursive space decomposition, which induces a geometric clustering of the vertices, and in fact it also induces a graph-theoretic clustering. This graph-theoretic clustering is then used in a force-directed algorithm, and this in turn improves the graph-theoretic clustering. Iterating this process improves both the drawing and the clustering; this process can be useful in applications.

The method was implemented in two and three dimensions using quad-trees and oct-trees [83]. Similar types of decomposition trees can also be used.

The claimed running time to compute the forces (that is, on one level) is approximately $O(|E| + |V| \log |V|)$. Example outputs are illustrated in Figure 20.2 [83].

1.3 A high-dimensional approach

We now describe more recent methods by Harel and Koren [76] for drawing very large graphs using high-dimensional embedding. The main idea of this method is first to draw a graph in very high dimensions (say 50) and then to project the embedding into two or three dimensions.

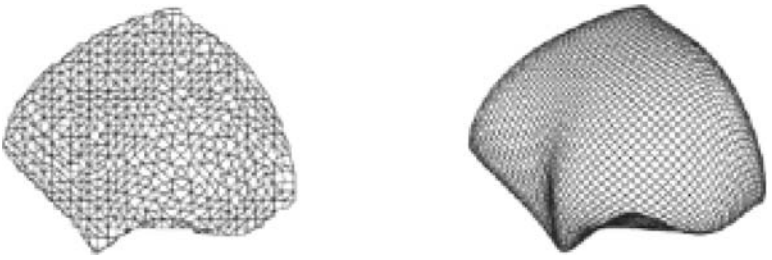

Figure 20.2. Graph with 2500 vertices on level 6 and the lowest level of the decomposition tree [83].

For the first step, drawing a graph in m dimensions, Harel and Koren choose m *pivot* vertices that are almost uniformly distributed on the graph, using an approximation algorithm of the *k-center* problem. Then the ith coordinate of each vertex is computed based on the graph-theoretic distance from the pivot vertex p_i using breadth-first search. This approach gives a rough but quick initial layout.

For the second step, they use *principal component analysis* (PCA) to choose a good projection of m-dimensional drawing into two or three dimensions. This method transforms a number of correlated variables into a smaller number of uncorrelated variables called *principal components*. The first principal component represents as much variability of the data as possible. Using only the first few principal components, PCA can reduce the dimensions of the data, maintaining the maximum possible variance (see [69]).

More specifically, Harel and Koren compute the first k eigenvectors of the *covariance matrix* (these correspond to the largest eigenvalues) using a simple power-iteration method [89]. Finally, they perform the projection using the direction of the eigenvectors. For details, see [76].

The claimed running time is $O(|V|+|E|)$, and the authors report that the method is rather independent of the structure of the graph (unlike the classical force-directed methods). They present experimental results with graphs of 10^5 vertices drawn in a few seconds, and 10^6 vertices drawn in a minute. Indeed, this method is much faster than the force-directed methods described in the previous sections.

In terms of the quality of the drawings, the method gives reasonable results. However, due to the limitations of linear projections, the 2D drawings have poorer quality compared with those produced by classical force-directed methods. Two sample outputs are illustrated in Figure 20.3 [76]. Figure 20.3 shows the drawing of the *crack* graph with 10,240 vertices and 30,380 edges; this drawing took 0.3 seconds.

For very sparse graphs such as trees, the method does not perform well in terms of the quality of the drawing. Harel and Koren also report that sometimes it is aesthetically better to choose different eigenspaces. The PCA method raises the possibility of creating graph drawing systems that browse views of the graph using different projections onto eigenspaces.

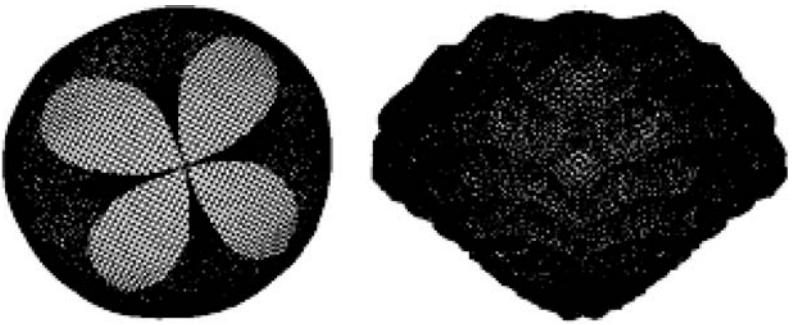

Figure 20.3. Drawings of (a) a 100×100 grid with opposite corners connected, and (b) the *crack* graph [76].

1.4 Spectral method

Spectral methods form part of the toolbox of algebraic graph theory [66] and have been used in many applications such as graph partitioning. The most widely used techniques use eigenvalues and eigenvectors of the *adjacency* matrix or *Laplacian* matrix of the graph.

The spectral graph drawing method was firstly introduced by Hall [73]. Recently, variations have been presented by a number of authors. We briefly review the main idea.

A simple spectral layout method that uses eigenvectors of the Laplacian matrix of a graph is described in [65,90]. Here, the eigenvectors are computed using a simple power iteration method. The layout method has been used for web-graphs and social network data. For details, see [90].

More sophisticated spectral methods for drawing large graphs are presented by Koren et al. [81, 82] and Koren [80].

ACE (Algebraic multigrid Computation of Eigenvectors) constructs a drawing of a graph using eigenvectors of the Laplacian. More specifically, the problem is reduced to minimizing a quadratic energy function, which can be expressed as a generalized eigenvalue problem. They authors present a very fast method for minimizing Hall's energy function [73] using a multiscale approach. For details, see Koren, Carmel, and Harel [81, 82].

It s claimed that ACE can draw graphs with 100,000 vertices in about 2 seconds and graphs of millions of vertices in a minute. However, the authors report that the running time of ACE depends more on the structure of the graph than on the high-dimensional approach. Figure 20.4 shows some results. Figure 20.4a shows the drawing of a folded 100×100 grid with 10,000 vertices and 18,713 edges. Figure 20.4b shows the drawing of a graph with 4,970 vertices and 7,400 edges. Figure 20.4c shows the drawing of the *crack* graph, with 10,240 vertices and 30,380 edges. Figure 20.4d shows the drawing of the *4elt* graph, with 15,606 vertices and 45,878 edges. Figure 20.4e and 20.4f show the drawings of the *dwa512* graph, with 512 vertices and 1,004 edges, drawn using a different choice of eigenvectors.

Koren [80] further extends the spectral approach to graph drawing using *degree-normalized eigenvectors*, which have some aesthetic advantages. He presents an algorithm for computing the degree-normalized eigenvectors quickly. For details, see [80].

1.5 Remarks

We conclude this section with a summary and some remarks on future research directions.

In this section, we briefly discuss current research in graph drawing, concentrating on drawing very large undirected graphs. Several algorithms are available, and we can roughly divide them into two approaches: the fast force-directed methods combined with either a multilevel approach, graph theoretic clustering, or geometric clustering; and spectral approaches that use eigenvectors of matrices associated with the graph. The first approach can handle a few thousand vertices, and the second approach can handle millions of vertices.

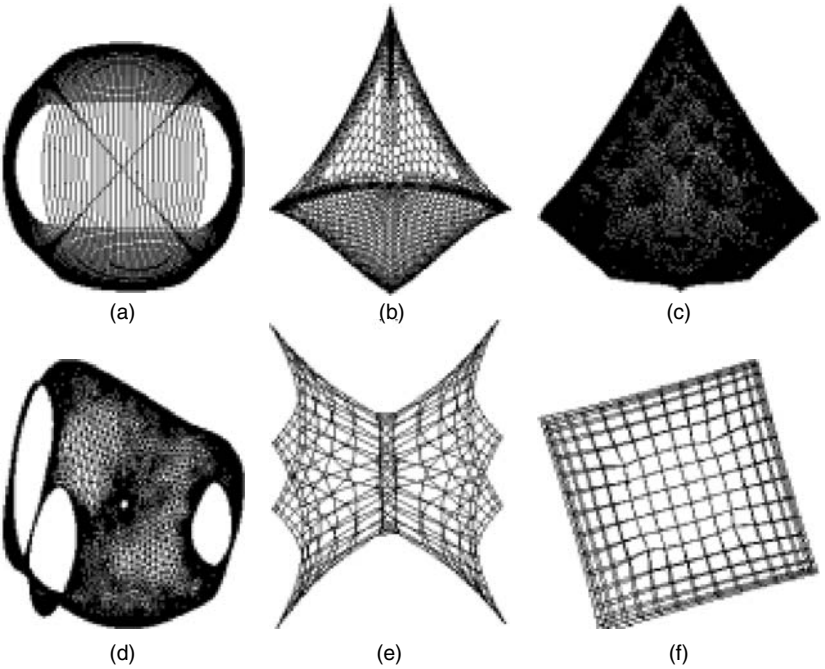

Figure 20.4. Examples of the drawings produced by ACE [81]. See text for details.

It should be noted that these new methods are not currently mature and need extensive evaluation. Some are beginning to be adopted into commercial tools, but at the time of this writing, commercial success has not been achieved. In particular, many authors present experimental results with sparse graphs and with regular structures, thus resulting in good quality drawing quickly. However, few of these methods have been evaluated with real-world data sets with dense graphs and irregular structure. With few exceptions, data such as webgraphs, PPI networks, and social network data have not been thoroughly tested.

Further, there has been no attempt to compare these methods formally. Hence, it would be interesting to conduct an extensive comparison of different approaches for drawing large graphs with real-world data sets.

Furthermore, it may be essential to modify these methods to produce a good-quality drawing of domain-specific data, since many networks and graphs in the real world exhibit special properties. For example, social networks [65] and biochemical pathways [84] exhibit special properties that may be exploited by constraints in force-directed methods. For the properties of special networks, see [64]. This is a challenging topic requiring further research.

Finally, users of real-world applications need good *navigation* methods to accompany the static visualization so that they can interact with the visualizations for further investigation based on their own interests or insights. Thus, future research also should include the design of good navigation methods that support efficient and effective *interaction* methods for the users. This area is also related to the *dynamic* visualizations of graphs, since graphs and networks in the real world are inherently dynamic and thus always changing.

2 NEW VISUALIZATION TOOLS AND TECHNOLOGIES

This section will discuss various visual technologies that are relatively inexpensive but are effective in increasing the accessibility and the amount of information being presented to a user on a screen. This section will also present current issues in utilizing and integrating such technologies to improve the capabilities of visualization.

2.1 High-resolution displays

Due to the marked advances of modern computing technologies, many commodity computer graphics hardware have achieved significant increases in performance and capability, including hardware accelerations of various three-dimensional computer graphics functions [33]. In 2004, many graphics cards commonly support a pixel resolution ranging from $1{,}024 \times 768$ pixels to $1{,}600 \times 1{,}200$ pixels, and further improvements are expected.

In the field of visualization (including both scientific and information visualization), the amount of data being displayed continues to increase, corresponding to the rapid progress of communication and computing technologies. Furthermore, these technological advances now allow scientists and engineers to push the boundaries of data analysis and simulation processes, resulting in the massive amount of data that needs to be visually inspected. In order to display such a large amount of information on a screen, the display system needs a large number of pixels.

Even with the modern commodity computer graphics hardware technologies, it is extremely difficult to provide a high-resolution display capability while maintaining real-time interactivities and three-dimensional complex geometry. In order to achieve super high-resolution displays with inexpensive commodity graphics hardware, Humphreys et al. introduced *WireGL* [35]. WireGL allows *OpenGL* rendering commands to be distributed across a cluster of inexpensive commodity graphics cards. This technology was rolled into a new project named *Chromium* [35]. The novel improvements in Chromium were (1) to provide a mechanism not requiring a user to execute an OpenGL application without modifications, and (2) to introduce a *Stream Processing Unit* (SPU) structure. Once a stream of OpenGL-rendering commands is intercepted by the core module of WireGL, these commands are distributed across a network of graphics hardware by the SPU.

Many distributed rendering or tiled display systems have been developed based on chromium technologies [36–41]. For example, NCSA at the University of Illinois [36], the Visualization Group at the Pennsylvania State University [37], and the *ViSLAB* at the University of Sydney [38] (see Figure 20.5) have developed and packaged chromium-based tiled display systems for scientific visualization. The *VIEWS* development group at LLNL has developed a parallel rendering system using chromium as well as Distributed Multi-headed X (DMX) technologies. With this system, not only the 3D graphics rendered by OpenGL but also other X-windows' widgets can be rendered in a distributed fashion [39].

Figure 20.5. A large tiled display being used in scientific visualization (ViSLAB, The University of Sydney).

Although chromium technology successfully provides distributed-rendering/tiled-display capability, it heavily relies on a fast local network (such as Gigabit Ethernet or Mirinet). Since chromium requires the transmission of many primitive geometry objects over the network, there would be a bottleneck if it were deployed over a cluster of computers on a LAN or a low-bandwidth network. Furthermore, it would be very challenging to provide this distributed rendering service to remotely situated client machines rather than to a tiled display system directly connected to the chromium cluster. In response to this challenge, Bethel et al. have combined chromium technology with a multithreaded scene graph framework [40, 41]. By providing a parallelizable scene graph framework, the rendering process was accelerated by using scene-specific knowledge; this allows the system to reduce the number of geometry objects to be transmitted over the network.

2.2 Augmented displays

One approach to increasing the amount of information conveyed through visualization is to increase the number of visual variables. This approach includes increasing the number of pixels (as described above) and the complexity of pictorial icons used to represent the information. Another approach is to present such pictures within other more information-rich environments. When a piece of information is presented in a context, the contextual information could be used to enhance the original representation and to add extra pieces of information. As a result, the visualization presented in a certain contextual environment could provide more information than the pictures alone can present.

Augmented Reality [42–46] and *Mixed R*eality [47–50] use various information and computing technologies (such as computer graphics, real-time range findings, and human–machine interfaces) to seamlessly integrate the virtual and real-world environments. Research in these fields has been driven by the need to improve the

human interface of the computing facilities. The general objective of such research activities is to enhance the human–machine interfaces and user experiences by complementary combinations of the real-world environment and the information/computing technologies.

When visualization of some datasets is required, a user is engaged in inspecting and analyzing the data. The process of data analysis often involves other types of information (such as paper documents, real-world experimental items, and conversational information among colleagues through intense collaboration), all of which exist outside visualized data spaces. Augmented display systems allow visualized information and tangible real-world items to coexist in the same user interaction spaces.

The *Digital Desk* uses a physical desk surface as a projection screen, allowing a user to inspect and interact with the visualized information and the physical object side by side. All user interactions are detected by various sensors, such as stroke sensors and a passive camera, and are communicated back to the visualized information [42]. Feirer et al. developed a knowledge-based augmented reality named KARMA [43]. This system utilized a see-through type *Head Mounted Display* to merge the visualization and real-world spaces.

The above Augmented Reality systems intend to place the visualized information in an environment along with real-world entities. The Mixed Reality system, on the other hand, attempts to place the information, typically images, of the real-world entities in the visualized space. For example, Kanade's *Virtualized Reality* system obtains three-dimensional information about real-world objects through multiple-camera passive range finders, and then places the 3D information in the virtual information space [49].

The Augmented and Mixed Reality systems mentioned above rely on a single visual display such as a desktop surface display or a head-mounted display. The visualization and interaction spaces are usually defined and constrained by this single display device. In order to increase the availability and accessibility of the visualized information and to free a user from spatial constraints, the *Everywhere Display* project at IBM uses the *Multisurface Display Projector* to turn nontethered surfaces into interactive display surfaces [46]. This type of system makes visualized data available and accessible anywhere and anytime, and has great potential to significantly improve how a user interacts with the visualized information.

2.3 Integration of visual technologies

The advances in information and communication technologies have resulted in many research projects that utilize them to create *Computer-Supported Collaborative Work* (CSCW) and *Computer-Mediated Communication* (CMC) in order to support local and remote collaboration. The marked improvements of various advanced visual technologies, as mentioned above, suggest that these new visually enabled technologies must be reevaluated and exploited as core technologies mediating the collaboration processes.

Many visually enabled CSCW or CMC systems [51–59] are designed based on the concept of *WYSIWIS* (What You See Is What I See) [60] and *WYSIWID* (What You See Is What I Do). Many such systems utilize large screen displays,

including workbench style displays and "natural" user interfaces [53–57], and some of them are commercially available. However, both the hardware and software of these systems are designed and developed to support *local* intense collaboration. Hence, they fall short in supporting remote intense collaboration. Moreover, many of these systems are still confined to wired input devices that are electro-mechanically or acoustically tracked. Some of the more natural user interfaces (such as *DiamondTouch* [55]) are touch-based interfaces, which require the screen to be touched before any tracking is possible.

The user interaction and communication models of such systems are, however, still based on Norman's gulf model [61]. According to this model, there are two information-processing devices (a computer and a user) connected to each other. The results of computation are passed on to a user via visual and audio output devices. The information from a user is transmitted to the computer through input devices such as a keyboard and mouse.

Norman explained various difficulties of using such systems based on the concept of a "gulf," which prevents a smooth transition between these two information-processing units. This model is useful in explaining the conventional interaction within computing systems. However, in real life, we interact with external objects and pieces of information in a more direct manner. Moreover, when networked interactive systems mediate intense collaboration between multiple parties, a new gulf is introduced between those systems. Therefore, the development of more intuitive user interfaces based on direct manipulation with the help of pervasive user input devices is the major challenge to developing a better visually enabled CSCW system.

Figure 20.6 illustrates a collaborative access table (*CAT*), which uses a passive-range, finder-based natural user input device. A horizontal display, for the computer output, is on the table surface. Cameras capture images of the hands and face of the user, who stands or sits on the left. The system utilizes a stereo range

Figure 20.6. A prototype of a CAT (built by Takatsuka, Eades, and students at the University of Sydney).

finder in order to track the user's hands in a 3D space. By tracking the user's hand and fingertip in the 3D space, the system replaces the mouse clicking function with a simple tapping action on the display surface. In this manner, a user does not have to learn other hand gestures in order to interact with information on the screen.

The uses of various communication-related computing technologies have been studied as the medium of intense local and remote user interactions. A number of computer-based technologies are available to support such systems (such as email, www, on-line chat, and video conferencing). Most of these technologies have been used simply to connect remotely located systems [52]. Hence, disparate users still have to interact with shared information through completely disparate user interaction spaces. The question is whether such technologies are effectively used to provide seamless collaboration. Without careful design and appropriate evaluations, such systems could add an extra "gulf" rather than filling the "gulf" of remoteness.

A clear understanding of the mechanisms of remote intense collaboration and the establishment of a computational framework based on shared visualized information to support the collaboration are the challenges in this field. The successful development of such a visually enabled collaboration system could enhance conventional office management, as well as research management/collaboration, and could help other research partners better understand each other's activities.

3 BEYOND THE VISUAL: MULTISENSORY DISPLAY

While the majority of work with abstract data displays has focused on the visual sense, there is also increasing interest in displaying abstract data across a wider range of human senses. Many data sets are characterized by their large size and multiattributed nature. By employing multisensory feedback, the goal is to widen the bandwidth between human and computer. With multisensory interfaces, the user can potentially perceive and assimilate multiattributed information more effectively (see Figure 20.7). By mapping different attributes of the data to different senses, such as the visual, auditory, and haptic (touch) sense, it may be possible to better understand large data sets.

This section will consider the display of abstract data using the alternative senses of audition and haptics. This introduction will progress to a consideration of integrating visual, auditory, and haptic displays. The section will then conclude with a discussion of the difficult issues encountered when designing multisensory displays.

3.1 Sound displays

Auditory displays can use sound parameters such as pitch, duration, timbre, and loudness to convey information to the user [1]. All these sound parameters can be controlled in the sound-generation process. The auditory sense is less adept than vision at localizing the position of sounds in space [2]. Nonetheless,

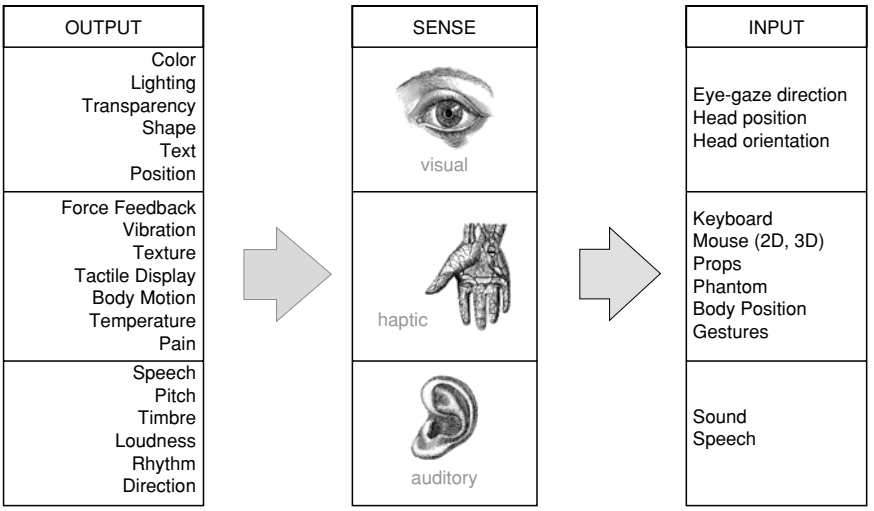

Figure 20.7. Multisensory user interfaces enable a number of different channels of input and output between the user and the computer.

the position of sound is a further parameter that can be used when designing information displays (see Figure 20.8).

A three-dimensional sound display can be achieved in two ways. One approach uses a spatially distributed array of speakers to generate what is called a *sound field simulation* [3]. The alternative approach is called *perceptual synthesis*. In this approach, the synthesized sound can be displayed on simple hardware such as a pair of headphones or loudspeakers. However, perceptual synthesis also requires an appropriate model of the user's head and ear shape, called the *Head-Related Transfer Function*. These functions incorporate the human perceptual

Figure 20.8. A user predicts stock market direction using a combined 3D visual and spatialized auditory display of stock market data [32].
Photo courtesy of CSIRO, Mathematical and Information Science, Canberra.

cues for sound localization into a source signal [3]. The models are complex, user specific, and difficult to generate [4].

The evolving field of study that focuses on displaying abstract data using sound is called *Information Sonification*. The term *information sonification* implies a mapping from the data attributes to the sound parameters [2]. When there is no such mapping, the term *audification* is used. *Audification* describes the direct playing of data as sound [2]. A good example of audification is the playing back of seismic events recorded from an earthquake [5].

In some sample applications of information sonification, sound has been used to assist in debugging software [6], to display scatter plots [7], to help understand parallel program performance [8], and to display computational fluid dynamics data [9].

Sound displays have also been combined with visual displays. For example, auditory signals based on a geiger-counter metaphor were used to display attributes of data collected from a petroleum well [10]. The user of this system could probe attributes of the well data with a sound tool while viewing a visual model of the petroleum well. Sound has also been used to display physiological parameters such as respiratory rate, body temperature, and heart rate, in conjunction with a visual readout of the same data [11].

3.2 Haptic displays

The word *haptic* derives from the Greek and means *to grasp*. The sense of touch differs from vision and hearing in that it relies on action from the user to generate the stimuli. For example, a person must tap against a surface to feel its hardness or move a hand across a surface to feel its texture.

In the real world, the haptic sense is typically used for exploring and handling objects. Exploration tasks involve the extraction of object properties such as shape, mass, and texture and also provide a sense of contact, position, and motion. Handling tasks are dominated by user motor actions such as grasping and object manipulation. For the user, haptic actions require a synergy of sensory exploration and motor manipulation [12].

Direct contact and displacement of the skin with an object provides tactile information, commonly described as touch. However, the human haptic system senses both tactile and kinesthetic information when touching an object [13]. Kinesthetic information provides the sense of position and motion of our limbs and joints. Current tactile displays are inadequate for use in real applications; however, it is possible to integrate force-feedback displays into current virtual environment systems [12]. For example, many platforms use the commercially available Phantom™ force-feedback device [14]. These displays can mimic a range of haptic sensations that the user senses through a combination of tactile and kinesthetic receptors.

The term *information haptization* is used when the sense of touch is used to display abstract data. The term *information tactilization* has also been suggested [15]. The word *haptic* refers to both the *tactile* and *kinesthetic* components of touch. Since most interactions involving the sense of touch rely on a combination of both tactile and kinesthetic feedback, the term *information haptization* is more general.

Although information haptization is a very new domain, some interesting applications have been developed. One of the first uses of force to display information was the *GROPE* project at the University of North Carolina [16]. This display was designed to assist users in molecular docking studies. Haptics has been used to display soil properties such as density, cohesion, and angle of internal friction by allowing the user to move a simulated plough blade through various sandy soils [17]. Force feedback was used to display a small set of properties such as static friction and surface deviations [18]. This approach allowed the user to feel surface textures on simulated surfaces. For example, in this way, different grades of sandpaper can be simulated. In turn, haptics properties such as surface texture, momentum, and compliance have been used to display attributes of stock market data [19] (see Figure 20.9).

In the stock market application, a haptic display is combined with a visual display. The same approach was also used in a control interface developed for a scanning probe microscope [20]. In this application, the user can feel the height and friction of the surface. As well as receiving this haptic information, the user also receives visual data from the surface height and color. In another application, force was used to help seismic interpreters look for patterns in geophysical data [21]. In this system, force feedback helped the user feel subtle features in the seismic data.

3.3 Designing multisensory displays

The term *multisensory* implies that *"more than one sensory modality is used to display the environment"* [1]. If the goal of multisensory display is to *widen the human-to-computer bandwidth*, then it is important that we strive to display different data

Figure 20.9. The Haptic Workbench incorporates a 3D visual display with the Phantom force-feedback device [14]. This device allows the user to feel attributes such as price momentum on a stock market chart [19].

Photo courtesy of: CSIRO, Mathematical and Information Science, Canberra.

attributes to different senses. This type of display has been characterized as a *complementary display* [22, 23].

Designing *complementary displays* seems a simple enough goal, yet often the senses can interact. For example, using sound in conjunction with haptics can alter the perceived stiffness of a surface [24]. So when a *hard* sound is played on contact, the surface is reported as being harder than when a *soft* sound is played—despite the fact that, in each case, the same haptic model is used to represent the surface contact. Likewise, changing the visual representation of the object can alter the perceived haptic stiffness of a spring. Thick visual representations of a spring feel stiffer than thinner ones, despite the same force being required to compress the spring [12].

Given the problems with multisensory interactions, is it wise to focus only on the visual display of abstract information? After all, some suggest that vision is the dominant sense. While it is true that vision is highly detailed and well suited to comparing objects arranged in space, it is equally true that hearing is effective for monitoring sounds from all directions, even when the source of the sound is not visible. Touch, as has been shown, does equally well as vision at discriminating texture [25]. Morton suggests that haptic texture cues may be more perceptually prominent than visual texture cues when both sources of information are present [25].

Welch and Warren go further: *"The dominance of vision is wrong"* [26]. In fact, different senses are well suited for different kinds of tasks. The problem is that it is not altogether clear what types of abstract data to display to each sense. To address this issue, the designer of a multisensory display must consider the physiological, perceptual, and cognitive capability of each sense.

Understanding the physiology of each sense helps in understanding its performance capabilities and bandwidth. For example, the range of colors that the eye can see or the frequency of sounds that can be heard are limited by the underlying physiology. Perception is dependent on physiology, but multiple levels of neural processing also influence it. For example, the same wavelength of light can appear to be a different color depending on the background color [27]. This difference is a result of the way nerves from the visual receptor cells are organized rather than the actual physiology of the eye's receptors.

The influence of higher neural processes on sensory perception is a general principle and can also be illustrated with hearing and touch [28]. For example, two similar sound frequencies can sound the same and the ability to distinguish them may depend on the musical training of the listener [29]. When displaying a haptic surface with force feedback, the display can give the impression of objects with a soft surface if the display frequency is low [12].

Cognition issues are also important when designing a display to recognize patterns. For example, the haptic sense may not be as useful for remembering complex patterns as the auditory sense. Some users may be more adept at using a particular sense, especially as attention to any single sense can influence performance with that sense [30]. Apart from attention, expectations, context, and knowledge can all influence what we see, hear, and feel [30].

Apart from physiological, perceptual, and cognitive concerns, the designer of multisensory displays must also consider the tasks of the intended user, characteristics of the data themselves, and specifics of the intended display hardware.

Although some attempts have been made to better characterize the design of visual displays [15], auditory displays [31], and even multisensory displays [32], much more theoretical work still needs to occur in these areas.

4. CONCLUSION

This chapter has described some of the new algorithms and technologies for information display. In most cases, these are untested outside universities and research laboratories. However, it is clear that a number of them will eventually find their way into commercial tools. As more novel concepts in information display are invented and tested, the way that we perceive information will be changed.

REFERENCES

[1] R. Stuart (1996): The Design of Virtual Environments. New York, McGraw-Hill.

[2] G. Kramer (1994): An Introduction to Auditory Display. Auditory Display: Sonification, *Audification and Auditory Interfaces*. Addison-Wesley.

[3] M. J. Evans, A. I. Tew, J. A. S. Angus (1997): Spatial Audio Teleconferencing—Which Way is Better? *International Conference on Auditory Display,* Palo Alto, California.

[4] E. M. Wenzel, M. Arruda, D. S. Kistler, and F. L. Wightman (1993): Localization using nonindividualized head-related transfer functions. *Journal of the Acoustical Society of America*, *94*, 111–123.

[5] C. Hayward (1994): Listening to the Earth Sing. Auditory Display: Sonification, Audification and Auditory Interfaces. G. Kramer, Addison-Wesley, pp. 369–404.

[6] D. H. Jameson (1994): Sonnet: Audio-Enhanced Monitoring and Debugging. Auditory Display: Sonification, Audification and Auditory Interfaces. G. Kramer, Addison-Wesley, pp. 253–266.

[7] T. M. Madhyastha and D. A. Reed (1994): A Framework for Sonification Design. Auditory Display: Sonification, Audification and Auditory Interfaces. G. Kramer, Addison-Wesley, pp. 267–290.

[8] J. A. Jackson and J. M. Francioni (1994): Synchronization of Visual and Aural Parallel Program Performance Data. Auditory Display: Sonification, Audification and Auditory Interfaces. G. Kramer, Addison-Wesley, pp. 291–306.

[9] K. McCabe and A. Rangwalla (1994): Auditory Display of Computational Fluid Dynamics Data. Auditory Display: Sonification, Audification and Auditory Interfaces. G. Kramer, Addison-Wesley, pp. 327–340.

[10] S. Barass and B. Zehner (2000): Responsive Sonification of Well-logs. *International Conference on Auditory Display*, Atlanta, Georgia, USA.

[11] W. T. Fitch and G. Kramer (1994): Sonifying the Body Electric: Superiority of an Auditory over a Visual Display in a Complex, Multivariate System. Auditory Display: Sonification, Audification and Auditory Interfaces. G. Kramer, Addison-Wesley, pp. 307–326.

[12] M. A. Srinivasan and C. Basdogan (1997): Haptics in Virtual Environments: Taxonomy, Research Status, and Challenges. *Computer & Graphics 21*(4) 393–404.
[13] N. I. Durlach and A. S. Mavor, (ed) (1995): Virtual Reality: Scientific and Technological Challenges. Washington, D. C. National Academy Press.
[14] J. K. Salsibury and M. A. Srinivasan (1997): Phantom-Based Haptic Interaction with Virtual Objects. *Computer Graphics and Applications 17*(5), 6–10.
[15] S. K. Card, J. D. Mackinlay, and B. Shneiderman, (ed): (1999): Information Visualization. *Readings in Information Visualization.* San Francisco, California, Morgan Kaufmann.
[16] J. J. Batter and F. P. J. Brooks (1972): GROPE-1. IFIP '71.
[17] D. F. Green and J. K. Salsibury (1998): Soil Simulation with a PHANToM. *The Third PHANToM User's Group Workshop*, Cambridge, Massachusetts, USA, MIT.
[18] D. F. Green (1997): Texture Sensing and Simulation Using the PHANToM: Towards Remote Sensing of Soil Properties. *The Second PHANToM User's Group Workshop*, Cambridge, Massachusetts, USA, MIT.
[19] K. Nesbitt (2002): Experimenting with Haptic Attributes for Display of Abstract Data. *Eurohaptics 2002 International Conference*, Edinburgh, Scotland.
[20] A. Seeger, J. Chen, and R. M. Taylor (1997): Controlling Force Feedback Over a Network. *The Second PHANToM User's Group Workshop*, Cambridge, Massachusetts, USA, MIT.
[21] J. P. McLaughlin and B. J. Orenstein (1997): Haptic Rendering of 3D Seismic Data. *The Second PHANToM User's Group Workshop*, Cambridge, Massachusetts, USA, MIT.
[22] M. R. McGee, P. D. Gray, and S. A. Brewster (2000): Communicating with feeling. *First Workshop on Haptic Human–Computer Interaction.*
[23] L. Y. Pao and D. A. Lawrence (1998): Synergistic Visual/Haptic Computer Interfaces. *Japan/USE/Vietnam Workshop on Research and Education in Systems, Computation and Control Engineering.*
[24] D. E. DiFranco, G. L. Beauregard, and M. A. Srinivasan (1997): The Effects of Auditory Cues on the Haptic Perception of Stiffness in Virtual Environments. ASME Dynamic Systems and Control Division.
[25] A. H. Morton (1982): Visual and Tactile Texture Perception: Intersensory Co-operation. *Perception & Pyschophysics* 31, 339–344.
[26] R. B. Welch and D. H. Warren (1980): Immediate Perceptual Response to Intersensory Discrepancy. *Psychological Bulletin 88*(3), 638–667.
[27] J. Itten (1970): The Elements of Color. New York, USA, Van Nostrand Reinhold.
[28] R. Sekuler and R. Blake (1990): Perception. New York, McGraw-Hill.
[29] G. Kramer, B. Walker, et al. (1997): Sonification Report: Status of the Field and Research Agenda, Prepared for the National Science Foundation by members of the International Community for Auditory Display.
[30] E. B. Goldstein (1989): Sensation and Perception, Brooks/Cole.

[31] S. Barass (1997): Auditory Information Design. Computer Science. Canberra, Australian National University.
[32] K. Nesbitt (2003): Designing Multi-sensory Displays for Abstract Data, School of IT, University of Sydney.
[33] NVIDIA (2003): http://www.nvidia.com.
[34] ATI (2003): http://www.ati.com.
[35] G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett, and P. Hanrahan, (2001): WireGL: A scalable graphics system for clusters. *Proceedings of SIGGRAPH 2001*, 129–140, August.
[36] G. Humphreys, et al. (2002): Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters, *Proceedings of the 29th Annual Conference on Computer Graphics and interactive techniques*, ACM Press, New York, NY, USA.
[37] NCSA, Display Wall-in-a-Box. (2003): http://www.ncsa.uiuc.edu/TechFocus/Deployment/DBox/overview.html.
[38] PSU, The Pennsylvania State University, High Resolution Tiled Display Wall (2003) http://gears.aset.psu.edu/viz/facilities/displaywall.
[39] ViSLAB, The University of Sydney (2003) http://www.vislab.usyd.edu.au
[40] ASCI VIEWS Visualization project (2003) http://www.llnl.gov/icc/sdd/img/infrastructures.shtml.
[41] E. W. Bethel, et al. (2002): Combining a Multithreaded Scene Graph System with a Tiled Display Environment, *Proceedings of the 2002 IS&T/SPIE Conference on Electronic Imaging and Technology*, The Engineering Reality of Virtual Reality.
[42] E. W. Bethel, et al. (2003): Sort-First Distributed Memory Parallel Visualization and Rendering, *Proceedings of IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, pp. 41–50.
[43] P. Wellnerr (1993): Interacting with Paper on the Digital Desk, *Communications of the ACM*, *36*(7), 87–96.
[44] S. Feiner, B. Macintyre, and D. Seligmann (1993): Knowledge-based Augmented Reality, *Communications of the ACM*, *36*(7): 53–62.
[45] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre (2001): Recent Advances in Augmented Reality. *IEEE Comp. Graph. & App*, *21*(6), 34–47.
[46] A. Fuhrmann, et al. (1998): Collaborative Visualization in Augmented Reality, *IEEE Computer Graphics and Applications*, *18*(4), 54–59.
[47] C. Pinhanez (2001): Using a Steerable Projector and a Camera to Transform Surfaces into Interactive Displays, CHI 2001, March/April, 369–370.
[48] P. Milgram and F. Kishino (1994): A taxonomy of mixed reality virtual displays, *IEICE Transactions on Information and Systems (Special Issue on Networked Reality)*, E77-D(12): 1321–1329.
[49] P. Milgram and H. Colquhoun, Jr. (1999): A Taxonomy of Real and Virtural World Display Integration (Mixed Reality; (eds) Yuichi Ohta and Hideyuki Tamura), Ohmsha Ltd. & Springer-Verlag, pp. 5–30.
[50] T. Kanade, et al. (1999): Virtualized Reality – Digitizing a 3D Time-Varying Event As Is and in Real Time, (Mixed Reality; (eds) Yuichi Ohta and Hideyuki Tamura), Ohmsha Ltd. & Springer-Verlag, pp. 41–57.

[51] M. Hirose, et al. (1999): Building a Virtual World from the Real World (Mixed Reality – Merging Real and Virtual Worlds; (eds) Yuichi Ohta and Hideyuki Tamura), Ohmsha.
[52] R. A. May, II (1999): HI-SPACE: A Next Generation Workspace Environment, Washington State University, Department of Electrical Engineering and Computer Science, May.
[53] H. Ishii and N. Miyake (1991): Towards an Open Shared Workspace – Computer and Video Fusion Approach of Teamworkstation, *Communications of the ACM*, *34*(12), 37–50.
[54] S. Coquillart and G. Wesche (1999): The Virtual Palette and the Virtual Remote Control Panel: A Device and an Interaction Paradigm for the Responsive Workbench. In *IEEE Virtual Reality '99 Conference (VR'99)*, Houston.
[55] P. H. Dietz and D. L. Leigh (2001): DiamondTouch: A Multi-User Touch Technology. In *ACM Symposium on User Interface Software and Technology (UIST)*, pp. 219–226.
[56] B. Leibe, T. Stanner, W. Ribarsky, Z. Wartell, D. Krum, B. Singletary, and L. Hodges (2000): The Perspective Workbench: Towards Spontaneous and Natural Interaction in Semi-Immersive Virtual Environments. In *IEEE Virtual Reality 2000 Conference (VR'2000)*, pp. 13–20. New Brunswick, NJ.
[57] I. Rauschert, P. Agrawal, S. Fuhrmann, I. Brewer, H. Wang, R. Sharma, G. Cai, and A. MacEachren (2002): Designing a Human-Centered, Multimodal GIS Interface to Support Emergency Management. In *10th ACM Symposium on Advances in Geographic Information Systems (ACM GIS'02)*, Washington, DC, USA.
[58] A. F. Seay, D. Krum, W. Ribarsky, and L. Hodges (1999): Multimodal Interaction Techniques for the Virtual Workbench. In *Proceeding of CHI'99*.
[59] P. L. Schmalstieg, L. M. Encarnacao, and Z. Szalavar (1999): Using Transparent Props for Interaction with Virtual Table. In *Syposium on Interactive 3D Graphics (I3DG'99)*, Atlanta.
[60] R. Raskar, et al. (1998): The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays, *SIGGRAPH 98*, Orlando, Florida.
[61] M. Stefik, et al. (1986): WYSIWIS Revised – Early Experiences with Multi-user Interfaces, *CSCW'86*, pp. 276–290.
[62] D. A. Norman (1986): Cognitive Engineering (User Centered System Design; (eds) D. A. Norman and S. W. Draper), Lawrence Erlbaum Associates.
[63] J. Barnes and P. Hut (1986): A Hierarchical *O(nlogn)* Force-Calculation Algorithm: *Nature 324*(4), 446–449.
[64] S. Bornholdt and H. G. Schuster, (ed) (2003): Handbook of Graphs and Networks: From the Genome to the Internet. Wiley-VCH.
[65] U. Brandes and D. Wagner (2003): Visone -Analysis and Visualization of Social Networks: Graph Drawing Software: pp. 321–340. Springer Verlag.
[66] F. R. K. Chung (1997): Spectral Graph Theory: *CBMS Reg. Conf. Ser. Math.* 92. American Mathematical Society.

[67] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis (1999): Graph Drawing: Algorithms for the Visualization of Graphs: Prentice-Hall.
[68] P. Eades (1984): A Heuristic for Graph Drawing. *Congresses Numerantium 42*, 149–160.
[69] B. S. Everitt and G. Dunn, (1991): Applied Multivariate Data Analysis: Arnold.
[70] T. Fruchterman and E. Reingold (1991): Graph Drawing by Force-Directed Placement. *Software-Practice and Experience 21*(11), 1129–1164.
[71] P. Gajer, M. T. Goodrich, and S. G. Kobourov (2000): A Multi-dimensional Approach to Force-Directed Drawings of Large Graphs. *Proceedings of Graph Drawing 2000: Lecture Notes in Computer Science 1984:* pp. 211–221: Springer Verlag.
[72] P. Gajer and S. G. Kobourov (2002): GRIP: Graph Drawing with Intelligent Placement. *Journal of Graph Algorithms and Applications 6*(3), 203–224.
[73] K. M. Hall (1970): An r-dimensional Quadratic Placement Algorithm. *Management Science 17*, 219–229.
[74] R. Hadany and D. Harel (2001): A Multi-Scale Method for Drawing Graphs Nicely. *Discrete Applied Mathematics 113*, 3–21.
[75] D. Harel and Y. Koren (2002): A Fast Multi-Scale Method for Drawing Large Graphs: *Proceedings of Graph Drawing 2000: Lecture Notes in Computer Science 1984:* Springer Verlag: 183-196 (2000) (Journal version: *Journal of Graph Algorithms and Applications 6*(3), 179–202.
[76] D. Harel and Y. Koren (2002): Graph Drawing by High-Dimensional Embedding: *Proceedings of Graph Drawing 2002: Lecture Notes in Computer Science 2528:* Springer Verlag, pp. 207–219.
[77] M. Junger and P. Mutzel, (ed) (2003): Graph Drawing Software: Springer Verlag.
[78] T. Kamada and S. Kawai (1989): An Algorithm for Drawing General Undirected Graphs. *Information Processing Letters 31*, 7–15.
[79] M. Kaufmann and D. Wagner, (ed) (2001): Drawing Graphs: Methods and Models: Lecture Notes in Computer Science Tutorial 2025: Springer Verlag.
[80] Y. Koren (2003): On Spectral Graph Drawing: *Proceedings of COCOON 2003: Lecture Notes in Computer Science 2697*: Springer Verlag, pp. 496–508.
[81] Y. Koren, L. Carmel, and D. Harel (2002): ACE: A Fast Multiscale Eigenvectors Computation for Drawing Huge Graphs: *Proceedings of IEEE Symposium on Information Visualization (InfoVis) 2002:* 137–144
[82] Y. Koren, L. Carmel, and D. Harel (2003): Drawing Huge Graphs by Algebraic Multigrid Optimization. *Multiscale Modeling and Simulation 1*(4), 645–673, SIAM.
[83] A. Quigley and P. Eades (2000): FADE: Graph Drawing, Clustering, and Visual Abstraction: *Proceedings of Graph Drawing 2000: Lecture Notes in Computer Science 1984:* pp. 183–196: Springer Verlag.
[84] F. Schreiber (2002): High Quality Visualization of Biochemical Pathways in BioPath. *Silico Biology 2*(2), 59–73.
[85] K. Sugiyama (2002): Graph Drawing and Applications for Software and Knowledge Engineers: World Scientific.

[86] K. Sugiyama, and K. Misue (1995): Graph Drawing by Magnetic Spring Model. *Journal of Visual Languages and Computing 6*(3): 217–231.
[87] C. Walshaw (2000): A Multilevel Algorithm for Force-Directed Graph Drawing. *Proceedings of Graph Drawing 2000: Lecture Notes in Computer Science 1984:* pp. 171–182: Springer Verlag.
[88] C. Walshaw (2003): A Multilevel Algorithm for Force-Directed Graph Drawing. *Journal of Graph Algorithms and Applications 7*(3), 53–85.
[89] D. S. Watkins (1991): Fundamentals of Matrix Computations: John Wiley.
[90] B. Brandes and S. Cornelsen (2003): Visual Ranking of Link Structures. *Journal of Graph Algorithms and Applications 7*(2), 181–201.