Interactive Visualization of Graph Pyramids

Andreas Kerren

University of Kaiserslautern, Computer Science Department
P.O.Box 3049, D-67653 Kaiserslautern, Germany
kerren@acm.org

Abstract. Graph or image pyramids can be used for hierarchical partitioning of images. One related application area is the automatical determination of landscapes in Landscape Ecology. The visualization of graph pyramids facilitates studies about their structure, such as their vertex distribution or height in relation of a specific input image. Thus, a researcher can debug algorithms and ask for statistical information. Furthermore, it improves the better understanding of contraction processes or the role of vertex and edge attributes. DGCVis is an interactive visualization tool that supports several coordinated views, for example on graph pyramids, subpyramids, and level graphs. Most important properties and applications of this tool are described in this paper.

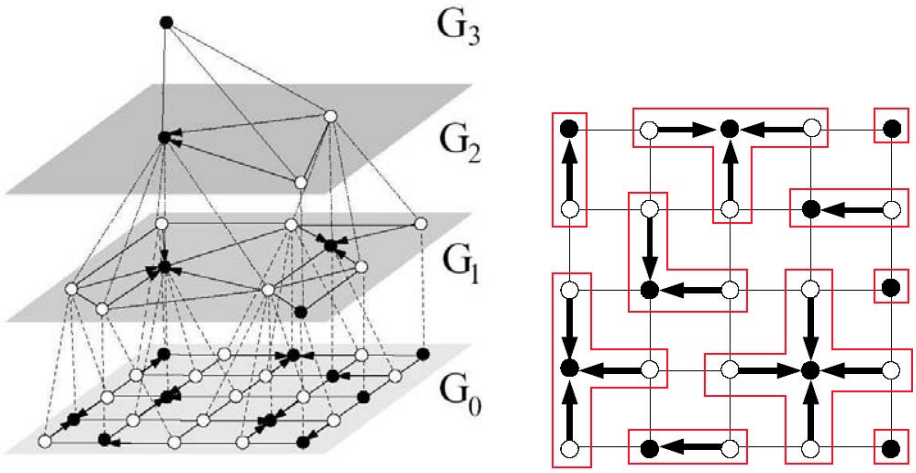
Keywords. Interactive Visualization, Human-Computer Interaction, Exploration, Graph Pyramids, Landscape Ecology

1 Introduction

Graph pyramids [1] (synonymous with *image pyramids*) store hierarchies of graphs and the linkage between consecutive graph levels. These data structures allow multiple scale and abstraction of the description of images. In general, they can be built from images or pre-segmented graphs in GML format [2] which are extended with any additional information. Each node of the base level represents a pixel with a specific color value. This is one of the node's attributes. The edges form a regular grid. In case of pre-segmented input graphs (a good overview of suitable image segmentation techniques is given by Pal & Pal [3] and Fu & Mui [4]), the base level corresponds to the elements of the input graph. Additionally, nodes and edges can be annotated with user-defined attributes, e.g., with colors or properties such as the degree of border precision between segmented image areas. Figure 1(a) shows a small example of a graph pyramid. In *regular* image pyramids, the proportion of the number of nodes at two levels l and $l+1$ is given by a constant reduction factor $r > 1$. This strict property is softened by *irregular* pyramids where the hierarchical structure is not known from the first.

By contracting edges within one level of the pyramid an algorithm, called *Dual Graph Contraction* (DGC) [1], computes the graph of the next higher level. Edge contraction is a process of collapsing the edge $e = (u, v)$ and melting

Dagstuhl Seminar Proceedings 05191
Graph Drawing
http://drops.dagstuhl.de/opus/volltexte/2006/341



(a) Small (irregular) graph pyramid (b) Contraction kernels

Fig. 1. Graph pyramid and contraction kernels are based on a 5×5 pixel image (images were taken from [5] and [6]). Directed edges represent the contraction processes, white nodes are collapsed in the next upper level, and dotted edges between the levels symbolize together with the connected nodes all contraction kernels.

the nodes u and v, i.e., one node (the surviver) remains and the other node is removed. The surviver adopts the edges that were incident to the removed node and its attributes are a combination of those from u, v, and e. DGC is specified by disjoint *contraction kernels* (see Figure 1(b)), each defining a possibly empty set of non-surviving nodes and the surviver. The selection of a contraction kernel depends on edge and node attributes. They are described by so-called *contraction rules* which are highly dependent on the application area. The visualization of these contraction processes is an important issue because it illustrates the core of this kind of *meta-segmentation*. See the article of Kropatsch [1] for further details on DGC algorithms and image pyramids.

There are several application areas of (irregular) graph pyramids. Image segmentation or feature detection are well-known examples [7,8]. Furthermore, Haxhimusa & Kropatsch [9] discuss the hierarchical partitioning of images using pairwise similarity functions on graph-based representations of images.

Another application is the automatical determination of landscapes and their properties in Landscape Ecology. Such properties can be very complex and interdependent, for example warmth, solar irradiation, soil texture, or humidity in a special area. A landscape itself can consist of several objects, such as lakes or forests. To determine the type of a landscape, we also need information about the size, the shape, as well as some context data of these landscape objects. Thus, a large forest without any streets in alpine scenery could be an interesting

wildlife habitat. In the project GEOGRAPH[1], the automatical determination of primitive landscape elements and types is done by segmentation on the basis of images (e.g., satellite images). Graph-based segmentation techniques use node and edge information to perform the meta-segmentation: Each object (landscape area) in a level can be regarded as node in a graph and edges between nodes indicate the neighborhood relations within that level. The identification of meaningful contraction rules is also part of the project and ongoing research. The result is a graph pyramid, which we can use as data structure for retrieving geographical information. Each pyramid level represents a kind of abstract map (land cover maps) of the input image: Lower levels describe more concrete maps and higher levels represent a higher level of abstraction. In the following, we call these maps *thematical maps* due to the possibility to generate them using additional information. The remainder of this paper is organized as follows: Next, our visualization approach is presented in Section 2. Then, Section 3 describes all supported graphical views and navigation techniques. An overview of the implementation of DGCVis is presented in Section 4. We conclude with Section 5 and give an outline of future work.

2 Visualization of Graph Pyramids with DGCVis

An appropriate visualization of graph pyramids and related information are important aspects to learn more about the pyramid structure, relationships between different levels, contraction processes, etc. In this context, the visualization of the whole graph pyramid is well suited for navigation purposes. All needed information is stored in this data structure and there is a clear and logical connection between their components. The visual representation of the pyramid in DGCVis can be seen as one more or less complex object that can be grasped and manipulated. Users can look at the pyramid from every possible viewpoint. They can move (pan) the pyramid, rotate it and zoom in and out of it using the mouse, see Figure 2.

In general, DGCVis supports two different node selection types: attribute selection and subtree selection. Both selection types exist side by side. Selected nodes are drawn with different colors (light green, yellow, or a mixture of both if a node is both subtree and attributes selected) depending on their selection types. For example, the user can select a special pyramid level and the visualization system opens a view with the represented thematical map computed by a down projection (cp. Section 3.4) to the base level, see Figure 2, left-hand side. So, the user can easily develop new contraction rules and verify them with the help of the visualization. He/she can analyze the use and computation of contraction kernels level by level as well as the resulting thematical maps. In addition, a good visualization tool should facilitate the understanding of

[1] This research has been supported by the Austrian Science Fund (FWF) under grant number P14662-N04.

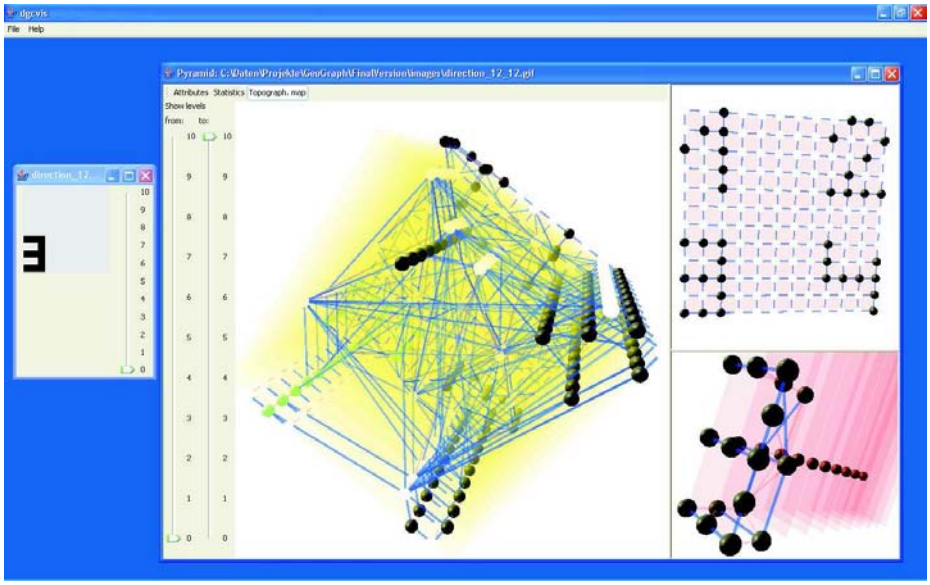


Fig. 2. Visualization of a Graph Pyramid (three different subviews, right window) and the result of a down projection (left window).

- the general structure of the pyramid itself,
- the contraction rule's effects on the pyramid structure,
- the verification of the contraction rule set and definitions,
- the correlation between pyramid levels and thematical maps,
- the role of node and edge attributes,
- related statistical information (e.g., its node distribution) and
- pyramid comparisons.

3 Graphical Views of DGCVis

DGCVis supports currently four different graphical views (2D and 3D) on the pyramid structure as well as related data. Each individual view was designed to fulfill different visualization needs. Note that the screenshot examples only show visualizations based on very small input data.

3.1 Graph Pyramid View (GPV)

This central 3D view presents three different subviews closely connected with each other. It can be used for common navigation purposes since graph pyramids are the central structures that store all information (see above). The graph nodes are visualized by colored spheres while the edges are thin cylinders, each

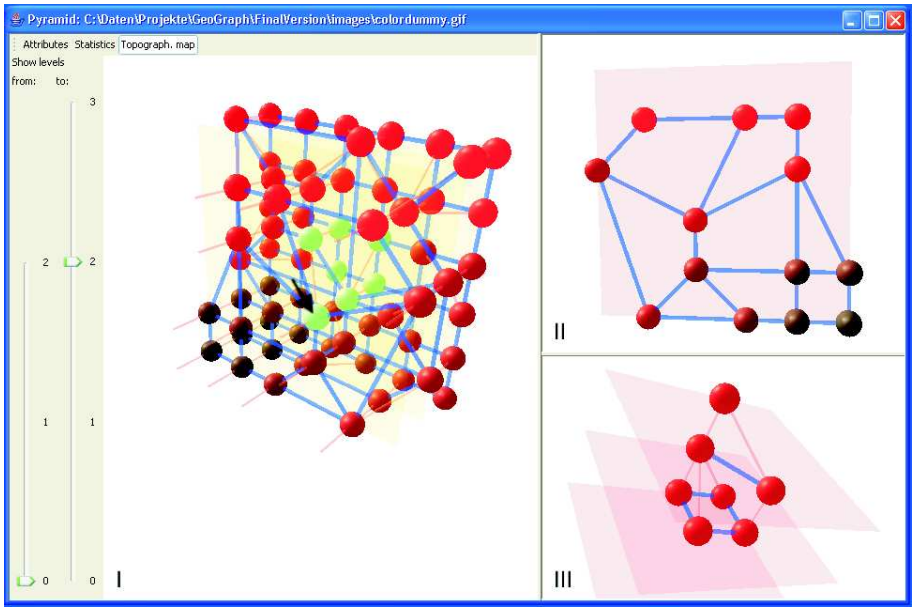


Fig. 3. Graph Pyramid View composed of Pyramid (I), Level (II), and Subtree (III) Subview.

connecting two spheres. The GPV represents a so-called Focus&Context technique (cp. [10] for example): users can focus on a specific level or subtree whereas the context of this focus is displayed in the Pyramid Subview.

Pyramid Subview: The Pyramid Subview displays the entire view of the data structure, i.e., each plane graph is drawn level by level. Light green lines which connect two graphs of different hierarchies represent the contraction kernels, see Figure 3 (Part I). This figure only shows a small example pyramid, bigger images produce bigger graph pyramids with much more visualization objects. In practice, the visual representation of a complete graph pyramid could be too complex for users to get deeper information. For this reason, our implementation supports a visualization of separate pyramid levels which are selected by choosing a level interval with the help of two sliders (left-hand side of Figure 3). This reduction of graphical objects suffices for the most practical application cases.

Level Subview: This part of the GPV shows the actual selected level, see Figure 3 (Part II). There are the same navigation facilities as in the Pyramid Subview. From the user's point of view, the visual comparison between the level graph and the thematical map visualization (see Subsection 3.4) is very important. Thus, he/she can see the mapping of the graph nodes and the appropriate image region of the computed thematical map.

Subtree Subview: The black arrow in the Pyramid Subview of Figure 3 (Part I) symbolizes a specific node selection at the third pyramid level. Then, the connected subtree is computed and placed in this third view. The subtree computation starts with the selected node and is linked via contraction kernels to the lower levels. These connections are traced and the resulting tree is copied into a new pyramid. In this way, the presentation of the whole data structure is reduced to the level, subtree, and node of interest as the reader can see in Figure 3 (Part III).

3.2 Attribute View (AV)

In the Attribute View, users can hierarchically navigate through a so-called *TreeView* [11] with all the attribute list types existing in the graph pyramid. Those attribute list types are grouped by level and then in the next hierarchy level by node or edge. In Figure 4(a), Nodes 29 and 31 were selected by the user. *Default RGB-A Attributes* is the name of the attribute list type which specifies the color of a node. Right now, we primarily work only with the color attribute because there are not any appropriate input examples yet. However, the system can work with an unlimited number of different attributes, such as humidity, population density, vegetation, etc. Through the use of a TreeView layout, we could develop a first prototype implementation of the AV very fast. As future aim, we would like to embed these attribute information into the GPV. But to do that, we have firstly to find better navigation techniques together with an advanced Focus&Context approach.

3.3 Statistics View (SV)

Global statistics are shown as chart diagrams within a separate window. At present, the statistical information a user can obtain are distributions of point nodes, face nodes, and quadedges, attribute list types of point nodes, face nodes, and quadedges, as well as contraction function sets in point and face graphs, each over levels. Figure 4(b) shows a distribution chart of point nodes/quadedges and two bar charts of attribute list types of face as well as quadedges. For our chart drawing, the free Java chart library JFreeChart [12] was used.

3.4 Thematical Map View (TMV)

This 2D view shows the selected level's or node's projection to the input image (called *down projection*) on which the graph pyramid is based. Among other things, attributes of each node on the selected level (for example the RGB value) or rather the selected node's attributes are forwarded to nodes of the base level along the tree paths built from the contraction kernels. In case of GML graphs as input, the down projection is performed with the help of a path per node for the specification of the area shape that is represented by the node. This information is part of the used GML specification. TMV contains a slider to

(a) AV (b) SV

Fig. 4. Attribute View (AV) and Statistics View (SV)

change the selected level, see the screenshot in Figure 5. This screenshot shows the down projection of the entire third level that corresponds to the input image, i.e., each node of the third level represents a connected color area of the input image. If there are only few selected nodes in the Pyramid Subview, then the TMV displays the down projection only on the basis of the selected nodes at the level selection. An example of this situation is shown in Figure 6: the Pyramid Subview was minimized (together with the selected node that was highlighted by a small black arrow in the Level Subview on the top right-hand side) to focus the down projection. The computed projection tree is displayed in the Subtree View and the nodes on the base level of the tree represent the red colored area in the TMV on the left-hand side. Note that the user has to rotate the objects in the GPV possibly in order to recognize such correlations.

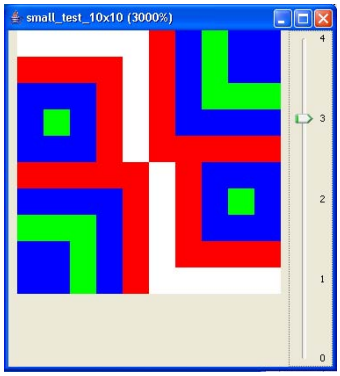


Fig. 5. Thematical Map View (TMV)

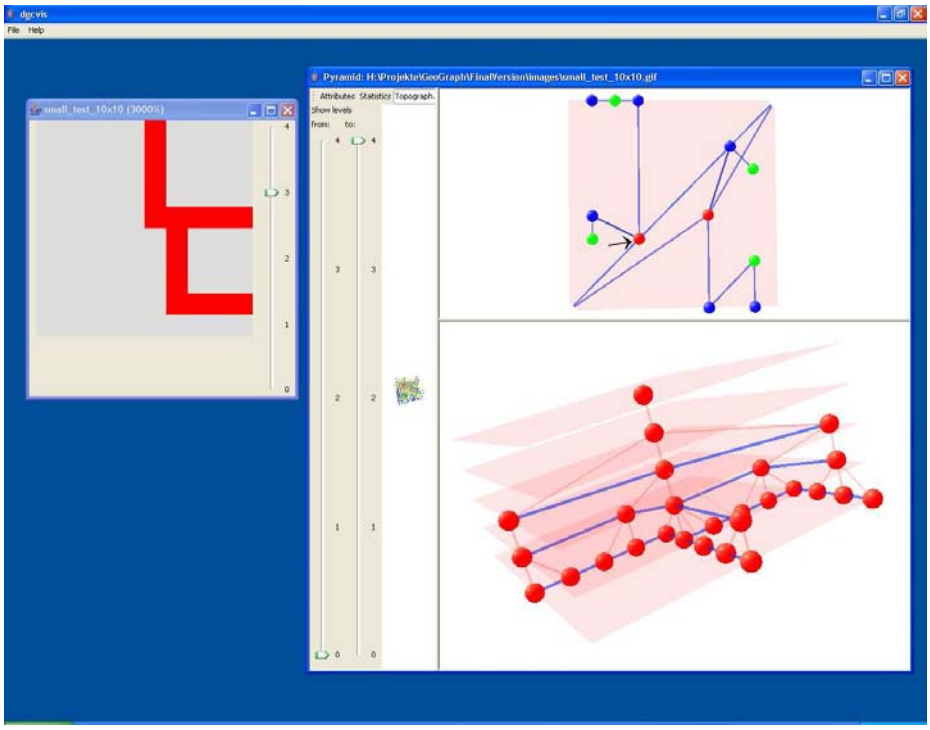


Fig. 6. Down Projection in the TMV

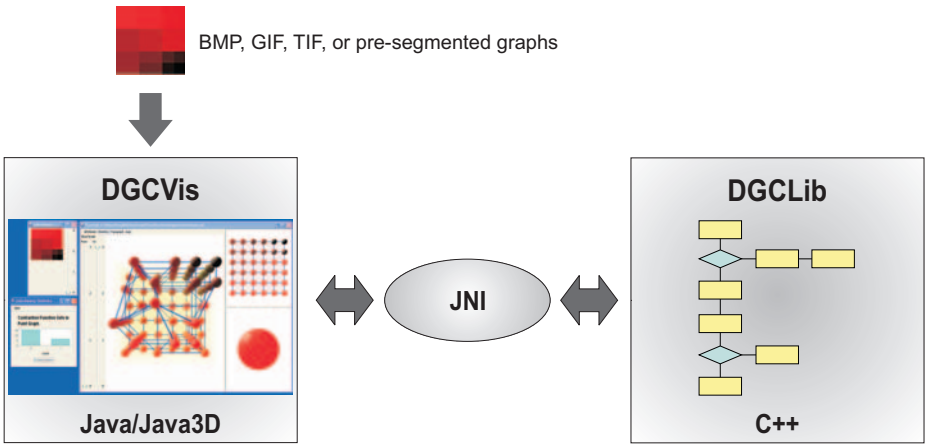


Fig. 7. Architecture

4 Implementation Aspects

DGCVis was implemented in Java [13,14] and Java 3D [15,16]. At first, we had to handle the problem that the computational part of our application is proceeded by a library, called DGClib, which performs all contraction processes and builds the entire graph pyramid. DGClib was implemented in C++ for performance reasons. There are several possibilities to transfer the data structure together with all references to our Java application for visualization. Due to the fact that platform independence was a requirement, we implemented two corresponding interfaces: Import data via JNI (Java Native Interface) and get data from a network interface via RMI (Remote Method Invocation). To get a clear distinction between the data structure and the Java-based visualization, the consequent implementation of a MVC (Model, View, Controller) model [17] was forced. As a consequence, there are no direct references between them. Figure 7 gives an overview of the architecture of DGCVis. More details about implementation and performance aspects can be found in paper [18].

5 Conclusions and Future Work

In this paper, a novel approach for the interactive visualization of graph pyramids was presented. The visualization covers several visualization needs of reseachers working on pattern recognition algorithms or on application areas. All implemented views offer many interaction and exploration possibilites to discover new correlations between contraction rules, thematical maps, and the graph pyramid's structure. As far as we know, our visualization tool is the first one in the area of interactive visualization of graph pyramids.

There are a lot of challenging problems in this area that we want to solve, e.g., interactive input of contraction rules from the visualization, visualization of

graph pyramid changes through movements of objects within an image sequence, or the visualization of pyramid comparisons. By clever visualization techniques and using pre-segmented input graphs, we could extensively avoid performance problems. However, large input images produce a huge amount of data to be visualized. Java3D has boundaries in this context and we have to look for other, more powerful solutions. From the viewpoint of Information Visualization, we have to find better navigation solutions and an advanced Focus&Context [19] approach. This will be one of our main research aims because an extensive study of the existing literature yielded no applicable results for this kind of problems. A good solution of these problems could also be transferred to other application areas.

Acknowledgements I would like to thank Florian Breier, Philip Kügler, and Michael Schreyer for implementing the DGCVis tool as well as the DGClib interface. Furthermore, I wish to thank the members of the Pattern Recognition and Image Processing Group at the TU Vienna for their useful ideas and support.

References

1. Kropatsch, W.G.: Building Irregular Pyramids by Dual Graph Contraction. IEE-Proc. Vision, Image and Signal Processing **142** (1995) 366–374
2. Himsolt, M.: GML – Graph Modelling Language. http://www.infosun.fmi.uni-passau.de/Graphlet/GML/ (2005)
3. Pal, N.R., Pal, S.K.: A Review on Image Segmentation Techniques. Pattern Recognition **26** (1993) 1277–1294
4. Fu, K.S., Mui, J.K.: A Survey on Image Segmentation. Pattern Recognition **13** (1981) 3–16
5. Haxhimusa, Y., Glanz, R., Saib, M., Langs, G., Kropatsch, W.G.: Logarithmic Tapering Graph Paramid. In Gool, L.V., ed.: Proceedings of the 24th DAGM Symposium 2002. LNCS 2449, Zürich, Swiss, Springer (2002) 117–124
6. Kropatsch, W.G., Haxhimusa, Y., Pizlo, Z., Langs, G.: Vision Pyramids that do not Grow too High. Pattern Recognition Letters **26** (2005) 319–337
7. Guigues, L., Herve, L., Cocquerez, J.P.: The Hierarchy of the Cocoons of a Graph and its Application to Image Segmentation. Pattern Recognition Letters **24** (2003) 1059–1066
8. Meer, P., Mintz, D., Montanvert, A., Rosenfeld, A.: Consensus Vision. In: Proceedings of the AAAI-90 Workshop on Qualitative Vision, Boston, MA, USA (1990) 111–115
9. Haxhimusa, Y., Kropatsch, W.G.: Hierarchical Image Partitioning with Dual Graph Contraction. In Michaelis, B., Krell, G., eds.: Proceedings of the 25th DAGM Symposium 2003, Magdeburg, Germany, Springer (2003) 338–345
10. Herman, I., Melançon, G., Marshall, M.S.: Graph Visualization and Navigation in Information Visualization: A Survey. IEEE Transactions on Visualization and Computer Graphics **6** (2000) 24–43
11. Walrath, K., Campione, M.: The JFC Swing Tutorial: A Guide to Constructing GUIs. Addison Wesley (1999)
12. Gilbert, D.: JFreeChart. http://www.jfree.org/jfreechart/index.php (2005)

13. Campione, M., Walrath, K., Huml, A.: The Java Tutorial Continued. 2. edn. Addison Wesley (1999)
14. Flanagan, D.: Java in a Nutshell. 3. edn. O'Reilly (2000)
15. Selman, D.: Java 3D Programming. Manning (2002)
16. Walsh, A.E., Gehringer, D.: Java 3D API Jump Start. Prentice Hall PTR (2002)
17. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series. Addison-Wesley, Reading, Massachusetts, USA (1995)
18. Kerren, A., Breier, F., Kügler, P.: DGCVis: An Exploratory 3D Visualization of Graph Pyramids. In Roberts, J.C., ed.: Proceedings of the 2nd International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV '04), London, UK, IEEE Computing Society Press (2004) 73–83
19. Card, S.K., Mackinlay, J.D., Shneiderman, B., eds.: Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann (1999)