Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/183-2
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: miglesias@cenatav.co.cu
ion@prip.tuwien.ac.at
krw@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-117 October 14, 2008

# Documentation for the Graph Pyramid Drawing Application [1]

*Mabel Iglesias Ham, Adrian Ion and Walter G. Kropatsch*

**Abstract**

The aim of this document is to support an easy orientation in the application developed so far for the pyramid drawing problem. This application is based on an algorithm that uses paths by means of the equivalent contraction kernels to draw the edges. The drawing shows a planar graph which preserves topology but also geometry of the original image. Also, it can deal properly with the presence of multiple edges and self loops which commonly appear in the top level of irregular pyramids. Using only straight lines, the self loops would disappear and multiple edges overlap. The functionality of detecting and drawing a set of generators in the top of the pyramid has been added, by means of computing a fake new level by a last contraction using a spanning tree, and finally reconstructing the remaining loops in the previous last level. For supporting the studies to measure new topological invariants the edges have been classified in contracted, removed and surviving edges using a code of colors. Details about the input text file, set of classes, and comments about future work have been included.

[1] Partially supported by the Austrian Science Fund under grants P18716-N13 and S9103-N13.

# Contents

# 1 Introduction

There is one published tool for the interactive visualization of graph pyramids [1]. It was designed to facilitate the studies about the structure of the pyramid and improve understanding of the contraction process using the visualization options. This algorithm uses straight lines, which do not show the whole information: self loops disappear, parallel edges collapse. On the other hand, there are applications where it is important to have a representation of a line drawing compressed, without loss of its geometric structure and topology.

There is a graph drawing community interested in finding high quality drawing algorithms to facilitate the visualization of complex relational networks. This means, for example, few edge bends in straight line drawings, orthogonal line segments for the edges, small display area, minimize crossings, good spatial and angular resolution, or recognizable symmetries. These methods find a distribution of nodes in the plane while we need to maintain a predefined position of nodes. In general, they do not deal with self loops and multiple edges, which occur in pyramids. The most prominent results can be seen in [2]. A novel algorithm for correctly visualizing graph pyramids preserving the geometry and the topology of the original image was presented in [3]. The software presented here is based on the mentioned algorithm and will aid future research in the field.

The aim of this document is to support an easy orientation in the software developed so far. The drawing of the edges follows a path by means of equivalent contraction kernels. It can deal properly with the presence of multiple edges and self loops which commonly appear in top level of pyramids. The functionality of detecting and drawing a set of generators in the top level of the pyramid has been added by means of computing a fake new level by a last contraction using a spanning tree, and finally reconstructing the remaining loops in the previous last level. For supporting the studies to measure new topological invariants the edges have been classified into contracted, removed and survived edges using a code of colors. Details about the input text file, set of classes and some comments about future work has been included.

The report is structured as follows. A brief recall about graph pyramids notions is presented in Section 2. An overview of the functionalities of the application is shown in Section 3. The used file format is described in Section 4. The structure of classes is presented in Section 5. Section 6 concludes with next steps and improvements, and gives an outlook of future functionality to be added.

## 2 Recall: Graph Pyramids

A *graph pyramid* $P$ is a stack of successively reduced graphs $P = \{G_0, \ldots, G_h\}$. Each level $G_k = (V_k, E_k)$, $1 \leq k \leq h$, is obtained by first *contracting* (*contraction process*) and then *removing* (*simplification process*) edges in the level $G_{k-1}$ below. Contracted edges of a level $G_{k-1}$ define trees called *contraction kernels* (CK) [4], whose vertices are merged to a single vertex in the level $G_k$ above. One vertex of each contraction kernel is called the *surviving vertex* and is considered to have been survived to the next level. Higher level vertices are related to the original input by *equivalent contraction kernels* (ECK) which, if applied, would achieve the same reduction in a single step. A path in a level $G_{k-1}$ is called a *connecting path* if it connects two (surviving) vertices $v, w$, and is made out of tree parts: a possible empty branch of the contraction kernel containing $v$, an edge called *bridge* that bridges the gap between two contraction kernels, and a possibly empty branch of the contraction kernel containing $w$. The base level bridge corresponding to an edge $e = (v, w)$ in a higher level $k$, is the bridge in the base level, connecting $ECK(v)$ and $ECK(w)$, and that has not been removed in any simplification up to the level $k$. The contraction process is controlled by the so called *decimation parameters* $(S_k, N_{k-1,k})$, where $S_k = V_k \subset V_{k-1}$ is the set of surviving vertices and $N_{k-1,k}$ are the contraction kernels. $(S_k, N_{0,k})$ denotes the surviving vertices and equivalent contraction kernels for contracting level 0 to level $k$.

Successive levels reduce the size of the data by a reduction factor $\lambda > 1$. Each level represents a partition of the base level into connected subsets of pixels. The construction of a pyramid is iteratively local. On the base level (level 0) of a pyramid the cells represent single pixels and the neighborhood of the cells is defined by the 4-connectivity of the pixels. A cell on level k + 1 (parent) is a union of neighboring cells in level k (children). Every parent computes its values independently of other cells on the same level.

A level of a dual graph pyramid consists of a pair $(G_k, \overline{G_k})$ of plane graphs $G_k$ and its geometric dual $\overline{G_k}$. The vertices of $G_k$ represent the cells on level $k$ and the edges of $G_k$ represent the neighborhood relations of the cells. The edges of $\overline{G_k}$ represent the borders of the cells on level $k$, including so called pseudo edges needed to represent neighborhood relations to a cell completely enclosed by another cell. Finally, the vertices of $\overline{G_k}$ represent junctions of border segments of $\overline{G_k}$. The sequence $(G_k, \overline{G_k})$, $0 \leq k \leq h$ is called (dual) graph pyramid (DGP). For more details and formal definitions of the terms defined in this section see for example [4].

# 3 Application Functionality

The main window of the application is shown in Fig. 1.

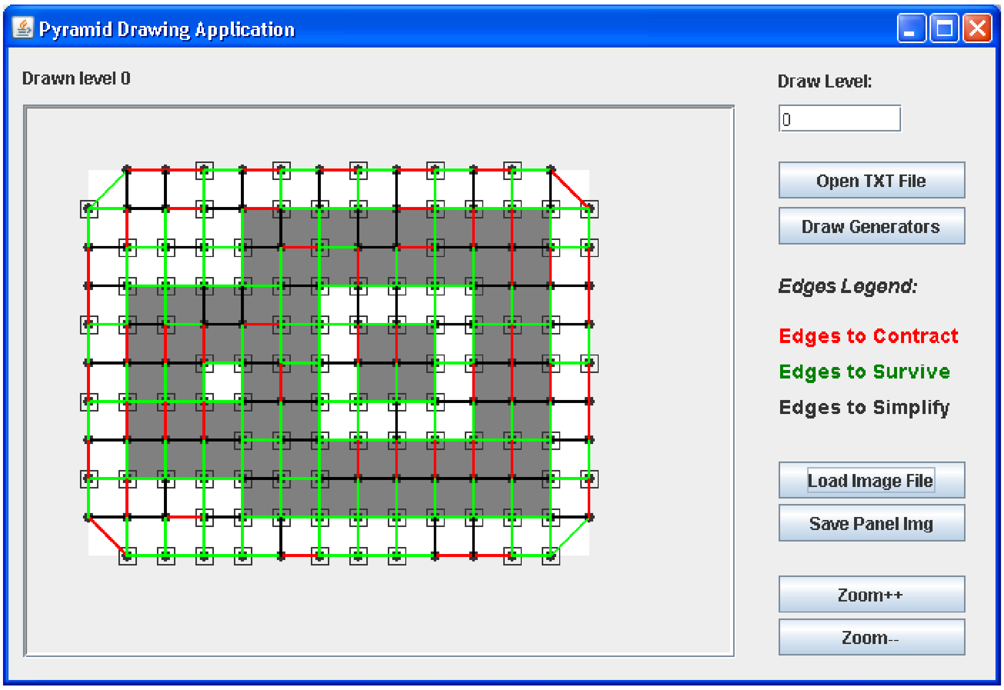


Figure 1: Main window of the application. Real image used to build the pyramid is shown.

The image shown in the window is the real image used to build the pyramid. This image is displayed as background and the level 0 graph is shown on top of it. The level of the graph that is being shown is specified in top of the drawing and also in the up right part of the window (Text: "Draw Level:"). Bellow, there are a number of buttons and labels which are described afterwards.

The software allows to read a text file with one pyramid information computed beforehand, and to draw each level using information of the *Contraction Kernels*. To draw a particular level of pyramid is enough to write the level number in the previously mentioned field and to press Enter. The button named *Open TXT File* is used to open the text file that contains the description of the pyramid.

In each level, the set of edges are drawn differentiating between the edges that will be contracted (red ones), the edges to be removed in the simplification process (black ones) and the ones that will survive to the next level (green ones). In the right part of the window this code of colors is described. The nodes that were selected as surviving ones in each contraction kernel are marked with a square as shown in Fig. 2.

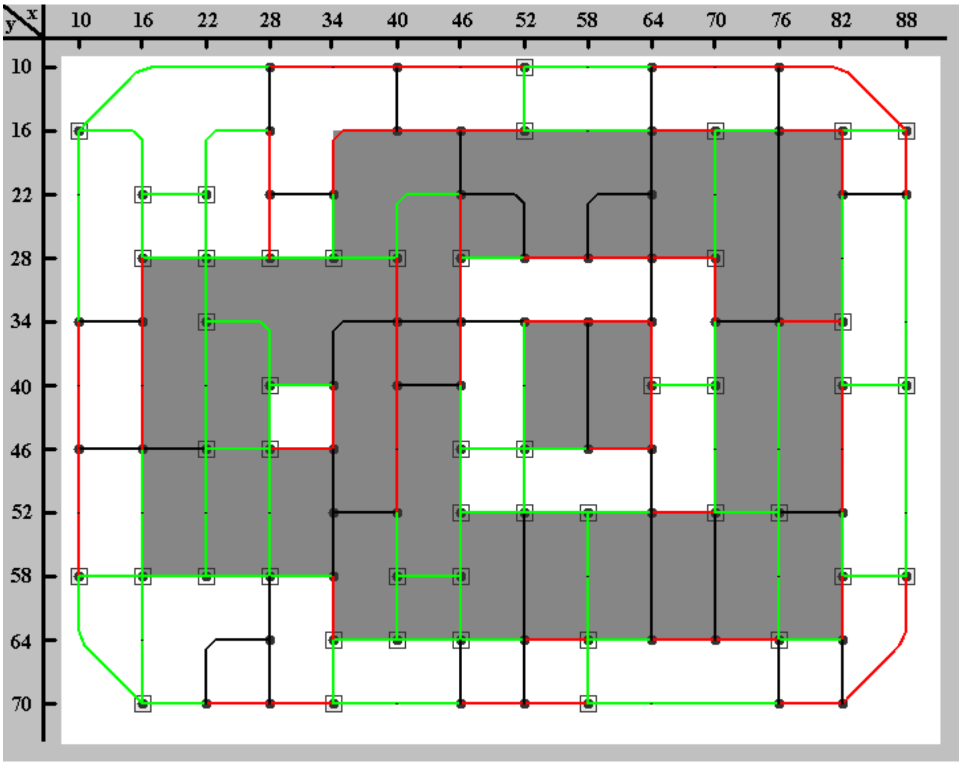


Figure 2: Level 1 of the pyramid. Each edge is drawn with different color. Contracted edges (red), Surviving edges (green) and Removed edges (black). Surviving nodes are marked with squares.

A new pyramid level has been added to obtain the generators. This new level is constructed by contracting the last level of the given pyramid by one spanning tree as shown in Fig. 3 in red lines.

This new graph (Fig. 4), is made only out of self loops. In this level, the generators of the homology groups are a subset of the self loops. There is an option to draw the generators in the previous level, where edges still fit on boundaries, using the button named *Draw Generators*. This button alternatively switches between *Draw Generators* and *Do Not Draw Generators*. When the option to *Draw Generators* is selected and the user is drawing the previous level, the generators are drawn as shown in Fig. 5.

Initially, the application only drew each level without the background image. Now, the user can select the background image using the button

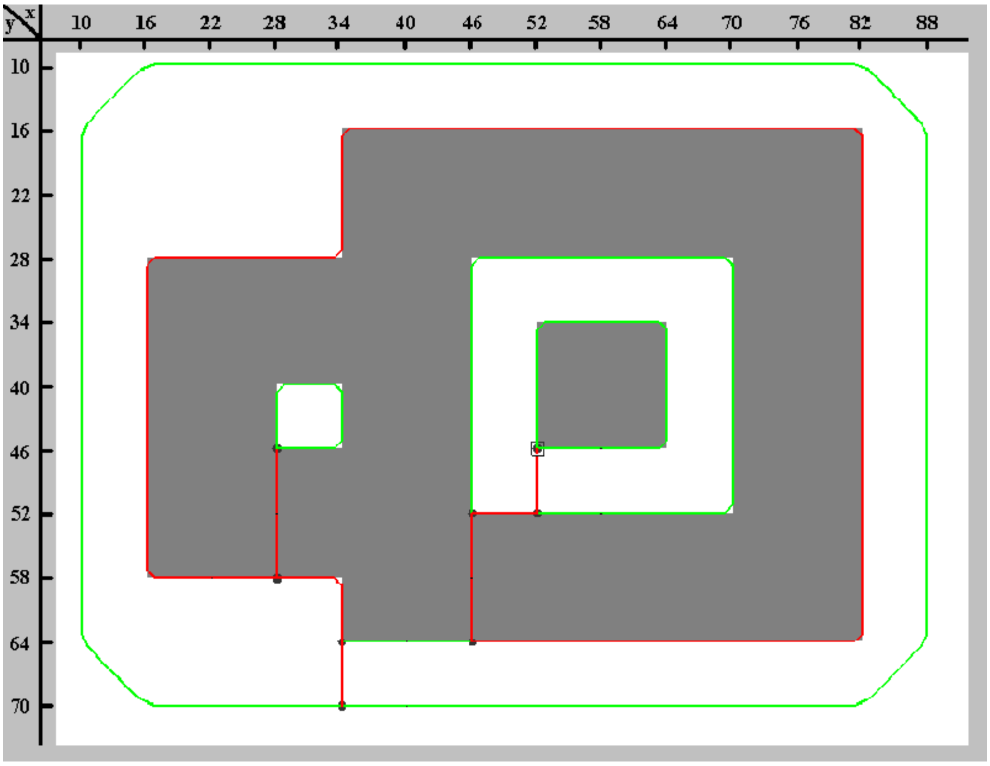


Figure 3: Red edges show the spanning tree computed in the last level of pyramid (Level 6). The edges in this tree will be contracted to the marked surviving node and the rest of the edges survive.

named *Load Image File* which will load the file and adjust the sizes to fit in the graph coordinates. Also, another button was added, named *Save Panel Image*, that allows to save a *.png* file of the image visualized in the window.

Finally, there are two more buttons to zoom in and out, that will allow seeing a more detailed part of a graph when it is bigger.

# 4 About the Input Text File

In the following the file format for the input graph pyramid is described. For a complete example see Section the Appendix.

% Comment lines are always preceeded with the percent symbol.

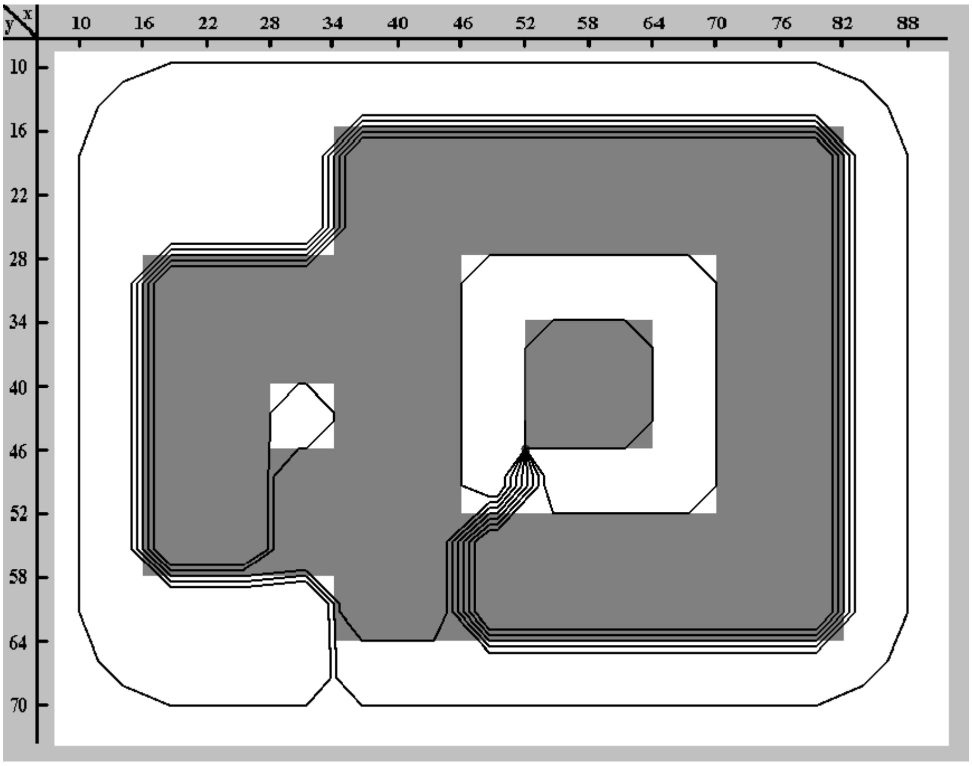


Figure 4: New level graph added on top of given pyramid, composed only of self loops and used to compute generators.

% Number of levels of the pyramid
7

% Number of Nodes of $G_0$, the graph representing the base level
% of the pyramid
150

% List of X Y coordinates
10 16
16 10
22 10
28 10

% ... more XY coordinates

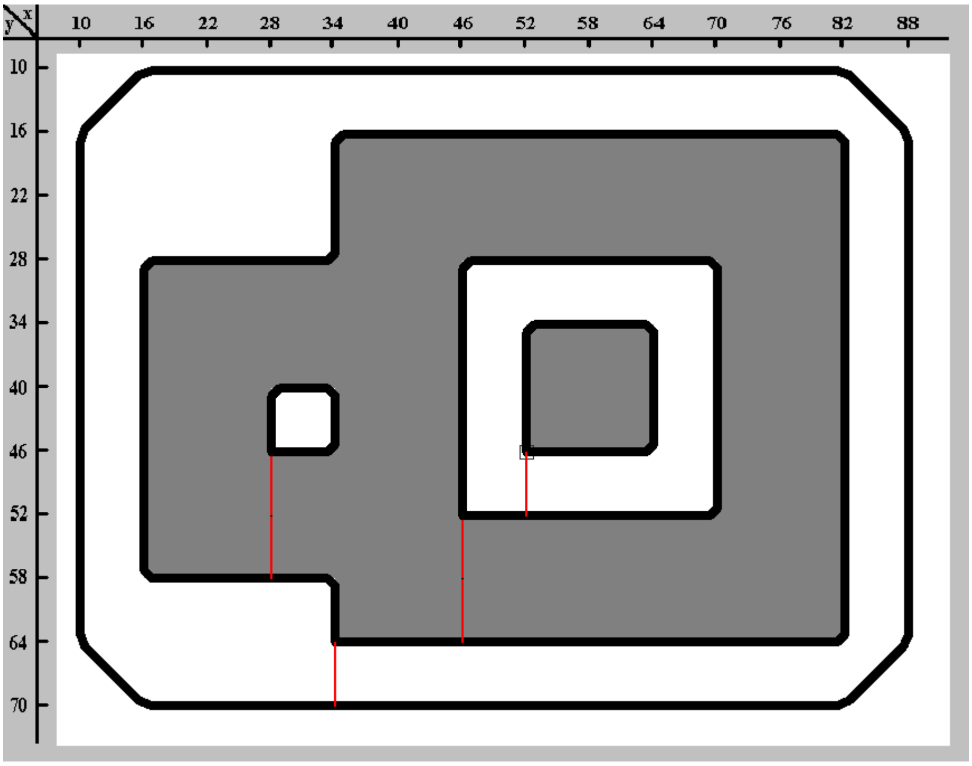


Figure 5: Set of generators are drawn with bold lines and are identified by the different cycles that they conform.

% Number of edges of $G_0$
279

% List of edges formed by three numbers describing an edge like:
% index of node1 in list of nodes of $G_0$, index of node2,
% and index of the corresponding edge in the list of edges
% of previous level graph (bridge).
% Comment: In first level the index of bridge = -1
0 1 -1
1 2 -1
2 3 -1
3 4 -1
4 5 -1

% ... more edges

% Number of contraction kernels of the level 0
98

% List of contraction kernels as:
% First number means the index of surviving node of the tree
% in the list of nodes from graph $G_0$ (always),
% then a list of indexes of the contracted edges
% in the list of edges from the actual graph

0
3 1 2
5 4
7 6
9 8
11 10
% more CK

% Number of edges of graph $G_1$
158

% List of edges with the same format than in previous level
% but with bridges different to -1
0 3 0
3 5 3
5 7 5
7 9 7
9 11 9

% ... more edges

% Number of contraction kernels of the level 1
47

% List of contraction kernels...

0
7 1 2
...

# 5 Structure of Classes, Implementation Details

The application has 11 classes:

1. PyramidsFrame
   %Main window
2. MyPanel
   %Drawing region within the main window
3. Pyramid
   %Pyramid description and functionality
4. Graph
   %Graph description for levels of pyramid
5. Node
   %Node description for graphs
6. Edge
   %Edge description for graphs
7. ContractionKernel
   %Contraction kernel(CK) description for pyramid information
8. TreeNode
   %Data structure for nodes in CK
9. SpanningTree
   %Spanning tree functionality for self loops level creation
10. SquarePoint
    %Aditional class supporting the drawing
11. Utils
    %General utility class

They are divided in the first 2 used by the main frame and the drawing panel, followed by the class that stores the structure of a pyramid (3). A pyramid structure contains a **Graph** (4) in each level that uses a definition for **Nodes** and **Edges** (5,6). Also, the pyramid uses a definition for **Contraction Kernels** on each level (7), which uses a tree structure with a node definition in the class **TreeNode** (8). For building the last level of the pyramid the **Spanning Tree** functionality has been defined in a separate class (9). Finally for supporting the drawing there is a definition for points that reaches an interconnection square (10) and a final class for general functionalities (11). The relations between the main classes are shown in Fig. 6.

The **pyramid class** has a method to load the pyramid information from a text file and to build the instance of the object. Also, there are other methods to compute all the steps of the drawing algorithm and to add the last level of pyramid (new contraction kernel by means of spanning tree and the last self loops made graph). The **SpanningTree** class also has the functionality of setting a flag for the edges that connects the final generators, in order to be visualized later. The whole sources have been commented.

# 6 Improvement and Next Steps

There are a lot of processes that can be computed in parallel as the computation of the Equivalent Contraction Kernel, measuring orders in the internal algorithm, etc. Java also provides very friendly classes to implements this, but, it is necessary to improve the memory demand of the actual application to be able to manipulate the set of new threads that will appear in the parallel running. In general the actual code can be optimized in time and space, and this will be done gradually giving more priority to further improvements in functionality so far.

The application will be enriched by a new functionality that will compute cohomology generators in the last level and also down project them to previous level till the base. For that, the application needs some more information in the input file as the edges of the corresponding face for each removed edge. This information can be added at the end of the line that describes and edge, maybe as a set of indexes of edges from the same level that bounds the face.

The application is planned to be extended to the visualization of 3D irregular pyramids, to support the computation of topological invariants in 3D too.

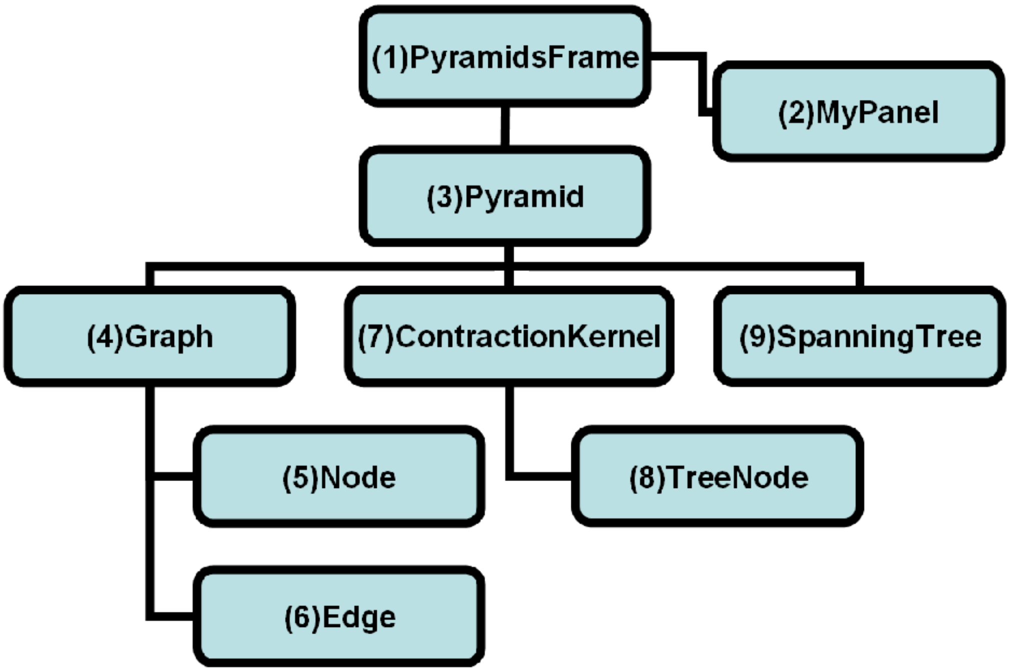


Figure 6: Relations between main classes in the application.

# References

[1] Kerren, A.: Interactive Visualization of Graph Pyramids. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, (2006) 1862-4405

[2] Duncan, C.A., Kobourov, S.G. and Sander, G.: Graph Drawing Contest Report. Proceedings of Graph Drawing Symposium. Volume 4875 of Springer Berlin /Heidelberg, (2007) 395-400

[3] Iglesias-Ham, M., Ion, A., Kropatsch, W. G. and García-Reyes, E. B.: Delineating Homology Generators in Graph Pyramids. Proceedings. Progress in Pattern Recognition, Image Analysis and Applications, 13th Iberoamerican Congress on Pattern Recognition, CIARP 2008, Havana, Cuba, September 9-12, Volume 5197 of Springer, LNCS, (2008) 576-584

[4] Kropatsch, W. G.: Building Irregular Pyramids by Dual Graph Contraction. IEE-Proc. Vision, Image and Signal Processing, Volume 142, No. 6, (1995) 366-374

# Appendix

In this section we are showing a complete example of an input text file for this application. The description on it refers to the graph pyramid that has been used in this technical report. Note that for a better organization of the document, this section uses double column style.

```
% Comment lines are always preceeded
% with the percent symbol.


% Number of levels of the pyramid
7


% Number of Nodes of G0, the graph
% representing the base level of the
% pyramid
150


% List of X Y coordinates
10 16
16 10
22 10
28 10
34 10
40 10
46 10
52 10
58 10
64 10
70 10
76 10
82 10
88 16
16 16
22 16
28 16
34 16
40 16
46 16
52 16
58 16
64 16
70 16
76 16
82 16
10 22
16 22
22 22
28 22
34 22
40 22
46 22
52 22
58 22
64 22
70 22
76 22
82 22
88 22
10 28
16 28
22 28
28 28
34 28
40 28
46 28
52 28
58 28
64 28
70 28
76 28
82 28
88 28
10 34
16 34
22 34
28 34
34 34
40 34
46 34
52 34
58 34
64 34
70 34
76 34
82 34
88 34
10 40
16 40
22 40
```

```
28 40
34 40
40 40
46 40
52 40
58 40
64 40
70 40
76 40
82 40
88 40
10 46
16 46
22 46
28 46
34 46
40 46
46 46
52 46
58 46
64 46
70 46
76 46
82 46
88 46
10 52
16 52
22 52
28 52
34 52
40 52
46 52
52 52
58 52
64 52
70 52
76 52
82 52
88 52
10 58
16 58
22 58
28 58
34 58
40 58
46 58
52 58
58 58
64 58
70 58
76 58
82 58
88 58
10 64
16 64
22 64
28 64
34 64
40 64
46 64
52 64
58 64
64 64
70 64
76 64
82 64
88 64
16 70
22 70
28 70
34 70
40 70
46 70
52 70
58 70
64 70
70 70
76 70
82 70
```

% Number of edges of $G_0$
279

% List of edges formed by three
% numbers describing an edge like:
% index of node1 in list of nodes
% of $G_0$, index of node2, and
% index of the corresponding edge
% in the list of edges of previous
% level graph (bridge).
% Comment:  In first level the index
% of bridge = -1
0 1 -1
1 2 -1

```
2 3 -1
3 4 -1
4 5 -1
5 6 -1
6 7 -1
7 8 -1
8 9 -1
9 10 -1
10 11 -1
11 12 -1
12 13 -1
0 14 -1
14 15 -1
15 16 -1
16 17 -1
17 18 -1
18 19 -1
19 20 -1
20 21 -1
21 22 -1
22 23 -1
23 24 -1
24 25 -1
25 13 -1
26 27 -1
27 28 -1
28 29 -1
29 30 -1
30 31 -1
31 32 -1
32 33 -1
33 34 -1
34 35 -1
35 36 -1
36 37 -1
37 38 -1
38 39 -1
40 41 -1
41 42 -1
42 43 -1
43 44 -1
44 45 -1
45 46 -1
46 47 -1
47 48 -1
48 49 -1
49 50 -1
50 51 -1
51 52 -1
52 53 -1
54 55 -1
55 56 -1
56 57 -1
57 58 -1
58 59 -1
59 60 -1
60 61 -1
61 62 -1
62 63 -1
63 64 -1
64 65 -1
65 66 -1
66 67 -1
68 69 -1
69 70 -1
70 71 -1
71 72 -1
72 73 -1
73 74 -1
74 75 -1
75 76 -1
76 77 -1
77 78 -1
78 79 -1
79 80 -1
80 81 -1
82 83 -1
83 84 -1
84 85 -1
85 86 -1
86 87 -1
87 88 -1
```

88 89 -1
89 90 -1
90 91 -1
91 92 -1
92 93 -1
93 94 -1
94 95 -1
96 97 -1
97 98 -1
98 99 -1
99 100 -1
100 101 -1
101 102 -1
102 103 -1
103 104 -1
104 105 -1
105 106 -1
106 107 -1
107 108 -1
108 109 -1
110 111 -1
111 112 -1
112 113 -1
113 114 -1
114 115 -1
115 116 -1
116 117 -1
117 118 -1
118 119 -1
119 120 -1
120 121 -1
121 122 -1
122 123 -1
124 125 -1
125 126 -1
126 127 -1
127 128 -1
128 129 -1
129 130 -1
130 131 -1
131 132 -1
132 133 -1
133 134 -1
134 135 -1
135 136 -1
136 137 -1
124 138 -1
138 139 -1
139 140 -1
140 141 -1
141 142 -1
142 143 -1
143 144 -1
144 145 -1
145 146 -1
146 147 -1
147 148 -1
148 149 -1
149 137 -1
0 26 -1
26 40 -1
40 54 -1
54 68 -1
68 82 -1
82 96 -1
96 110 -1
110 124 -1
1 14 -1
14 27 -1
27 41 -1
41 55 -1
55 69 -1
69 83 -1
83 97 -1
97 111 -1
111 125 -1
125 138 -1
2 15 -1
15 28 -1
28 42 -1
42 56 -1
56 70 -1

70 84 -1
84 98 -1
98 112 -1
112 126 -1
126 139 -1
3 16 -1
16 29 -1
29 43 -1
43 57 -1
57 71 -1
71 85 -1
85 99 -1
99 113 -1
113 127 -1
127 140 -1
4 17 -1
17 30 -1
30 44 -1
44 58 -1
58 72 -1
72 86 -1
86 100 -1
100 114 -1
114 128 -1
128 141 -1
5 18 -1
18 31 -1
31 45 -1
45 59 -1
59 73 -1
73 87 -1
87 101 -1
101 115 -1
115 129 -1
129 142 -1
6 19 -1
19 32 -1
32 46 -1
46 60 -1
60 74 -1
74 88 -1
88 102 -1
102 116 -1
116 130 -1
130 143 -1
7 20 -1
20 33 -1
33 47 -1
47 61 -1
61 75 -1
75 89 -1
89 103 -1
103 117 -1
117 131 -1
131 144 -1
8 21 -1
21 34 -1
34 48 -1
48 62 -1
62 76 -1
76 90 -1
90 104 -1
104 118 -1
118 132 -1
132 145 -1
9 22 -1
22 35 -1
35 49 -1
49 63 -1
63 77 -1
77 91 -1
91 105 -1
105 119 -1
119 133 -1
133 146 -1
10 23 -1
23 36 -1
36 50 -1
50 64 -1
64 78 -1
78 92 -1
92 106 -1

```
106 120 -1
120 134 -1
134 147 -1
11 24 -1
24 37 -1
37 51 -1
51 65 -1
65 79 -1
79 93 -1
93 107 -1
107 121 -1
121 135 -1
135 148 -1
12 25 -1
25 38 -1
38 52 -1
52 66 -1
66 80 -1
80 94 -1
94 108 -1
108 122 -1
122 136 -1
136 149 -1
13 39 -1
39 53 -1
53 67 -1
67 81 -1
81 95 -1
95 109 -1
109 123 -1
123 137 -1

% Number of contraction kernels of
% the level 0
98

% List of contraction kernels as:
% First number means the index of
% surviving node of the tree in the
% list of nodes from graph G_0
% (always), then a list of indexes
% of the contracted edges in the
% list of edges from the actual
% graph
0
3 1 2
5 4
7 6
9 8
11 10
13 12
16 15
18 17
19
20
22 21
23 242
24 252 253
25
27 152
28
29
30
32 31
35 34
38 263
39 272 273
41
42
43
44
45
46
47 213
48
49
50
54 144 145
55 155
56 165
```

```
59 56
60
61 215
62 225
63
64
65 255 256
66
71 175
72
73 196
74
77
78 246
80 266
81 275 276
82 147
83 157
84 167
85 177
86
88
89
90
91
100
101
102
103 218
104 228
105 238
106 248
107 258
108
110 149
111 159
112
113
114
115
116
122
123 278
127 119
128
129
130
131
132
133
134
135
136
138 130
139
140
141
143 135
144
145
148 139 140
149
```

% Number of edges of graph $G_1$

```
158


% List of edges with the same format
% as in the previous level but with
% bridges different to -1
0 3 0
3 5 3
5 7 5
7 9 7
9 11 9
11 13 11
0 27 13
18 19 18
19 20 19
20 22 20
22 23 22
```

23 24 23
24 25 24
25 13 25
27 28 27
29 30 29
32 47 32
38 39 38
41 42 40
42 43 41
43 44 42
44 45 43
46 47 45
47 48 46
48 49 47
49 50 48
54 55 52
56 71 54
59 60 57
60 61 58
61 62 59
62 63 60
64 65 62
65 66 63
71 72 68
73 74 70
77 78 74
80 81 77
82 83 78
83 84 79
84 85 80
85 86 81
88 89 84
89 90 85
90 91 86
100 101 95
102 103 97
103 104 98
104 105 99
105 106 100
106 107 101
107 108 102
110 111 104
111 112 105
112 113 106
113 114 107
115 116 109
122 123 116
128 129 121
129 130 122
130 131 123
131 132 124
132 133 125
133 134 126
134 135 127
135 136 128
138 139 131
139 140 132
140 141 133
141 143 134
143 144 136
144 145 137
145 148 138
148 149 141
149 123 142
0 54 143
54 82 146
82 110 148
110 138 150
27 41 153
41 55 154
55 83 156
83 111 158
111 138 160
16 28 162
28 42 163
42 56 164
56 84 166
84 112 168
127 139 170
3 16 171
16 29 172
29 43 173

```
71 85 176
85 113 178
113 127 179
127 140 180
18 30 182
30 44 183
59 72 185
72 86 186
86 100 187
100 114 188
114 128 189
128 141 190
5 18 191
32 45 193
45 59 194
59 73 195
73 101 197
101 115 198
115 129 199
19 32 202
32 46 203
46 60 204
60 74 205
74 88 206
88 102 207
102 116 208
116 130 209
130 143 210
7 20 211
61 89 216
89 103 217
103 131 219
131 144 220
35 48 223
62 90 226
104 132 229
132 145 230
9 22 231
22 35 232
35 49 233
49 63 234
63 77 235
77 91 236
91 105 237
105 133 239
23 50 243
50 64 244
64 78 245
78 106 247
106 134 249
11 24 251
24 65 254
65 107 257
107 135 259
135 148 260
25 38 262
38 66 264
66 80 265
80 108 267
108 122 268
122 136 269
136 149 270
13 39 271
39 81 274
81 123 277

% Number of contraction kernels of
% the level 1
47

% List of contraction kernels...
0
7 1 2
13 4 5 155
20 7 8 97
23 10
25 12 148
27
28
41 80 81
```

```
42
43 91 92
44
45 107 108 109
46 113 114 115
50 23 24 25 132 139
56
66 33
71
77 30 31 44 134 135
78
80 151
81
84
85 41 100
88
89
102
103
104
106 49
107
110 76 77
111
112
113
115
116
122 153
123 73 74
128 102 103
129
130
132 61
135 63 64
138
141 67 68 96
145 70 71
```

% Number of edges of graph $G_2$

```
73
```

```
% List of edges
0 7 0
7 13 3
0 27 6
20 23 9
23 25 11
25 13 13
27 28 14
41 42 18
42 43 19
43 44 20
44 45 21
46 50 22
56 71 27
71 85 34
77 78 36
80 81 37
84 85 40
88 89 42
89 77 43
102 103 46
103 104 47
104 106 48
106 107 50
110 111 52
111 112 53
112 113 54
113 128 55
115 116 56
122 123 57
128 129 58
129 130 59
130 132 60
132 135 62
135 122 65
138 141 66
141 145 69
145 123 72
```

```
0 110 75
110 138 78
27 41 79
41 111 82
111 138 83
43 28 84
28 42 85
42 56 86
56 84 87
84 112 88
71 85 93
85 113 94
20 44 98
128 141 104
46 45 106
45 115 110
115 129 111
46 88 116
88 102 117
102 116 118
116 130 119
7 20 121
77 89 122
89 103 123
104 132 128
132 145 129
23 50 138
50 78 140
78 106 141
66 107 145
107 135 146
25 66 149
66 80 150
80 122 152
13 81 156
81 123 157

% Number of contraction kernels of
% the level 2
29

% List of contraction kernels...
0 2
7
20 49
41
42
43 42
56 45
66 4 68 69
71
77
78 11 54 64
85
89
102
103
106 21
107
111
112
113
116 27 52
122
123 36 71 72
128
129
130
135 32
138 38
141
```

% Number of edges of graph $G_3$

```
44

% List of edges in level 3
0 7 0
7 123 1
20 66 3
```

```
41 42 7
42 43 8
43 20 9
56 71 12
71 85 13
77 78 14
89 77 18
102 103 19
103 106 20
106 107 22
111 112 24
112 113 25
113 128 26
122 123 28
128 129 29
129 130 30
130 135 31
135 122 33
138 141 34
141 123 35
0 138 37
0 41 39
41 111 40
111 138 41
43 42 43
42 56 44
56 112 46
71 85 47
85 113 48
128 141 50
116 129 53
78 102 55
102 116 56
116 130 57
7 20 58
77 89 59
89 103 60
78 106 65
66 107 66
107 135 67
66 122 70
```

```
% Number of contraction kernels of
% the level 3
12
```

```
% List of contraction kernels...
85 7
89 38
102 34 35 40
103
111
113 14 29
122 2 3 4 5 20 42 43
123
128
130 18
138 0 23
141
```

% Number of edges of graph $G_4$

```
18
```

```
% List of edges in level 4
138 123 1
89 89 9
102 103 10
103 102 11
111 113 13
113 128 15
122 123 16
128 130 17
130 122 19
138 141 21
141 123 22
122 111 25
111 138 26
85 85 30
85 113 31
```

```
128 141 32
102 130 36
89 103 39

% Number of contraction kernels of
% the level 4
10

% List of contraction kernels...
85
89
102
103
111 11
113
128
130
138 0
141
```

% Number of edges of graph $G_5$

```
15

% List of edges in level 5
89 89 1
102 103 2
103 102 3
111 113 4
113 128 5
128 130 7
130 111 8
138 141 9
141 138 10
111 138 12
85 85 13
85 113 14
128 141 15
102 130 16
89 103 17

% Number of contraction kernels of
% the level 5
8

% List of contraction kernels...
85
89
102
103
113
128
130 6
141 7
```

% Number of edges of graph $G_6$

```
12

% List of edges in level 6
89 89 0
102 103 1
103 102 2
130 113 3
113 128 4
128 130 5
141 141 8
85 85 10
85 113 11
128 141 12
102 130 13
89 103 14
```