Homology Group Generator Analysis in Irregular Graph Pyramids

Mabel Iglesias Ham[1,2], Adrian Ion[2], Walter G. Kropatsch[2], and Edel B. García[1] *

[1] Pattern Recognition Department, Advanced Technologies Application Center, 7a No. 21812, Siboney, Miramar, 12200, Havana, Cuba
{miglesias,egarcia}@cenatav.co.cu
[2] Pattern Recognition and Image Processing Group, Faculty of Informatics, Vienna University of Technology, Austria
{mabel,ion,krw}@prip.tuwien.ac.at

Abstract. Computation of homology generators using an irregular graph pyramid can significantly increase performance compared to the classical methods. First results in 2D exist and show the advantages of the method. The generators are computed in upper levels of pyramid where it is known that the graphs contains a number of self loops and multiple edges product of the contraction processes. Using a straight lines strategy to draw this edges would not be useful to analyze the graphs on those levels. This paper presents a novel algorithm for nicely visualize irregular graph pyramids, including multiple edges and self loops which preserves the geometry and the topology of the original image. This new algorithm is used to give new insights about the top-down delineation of homology generators in irregular graph pyramids.

Keywords: irregular graphs pyramids; pyramid drawing; homology group generators.

1 Introduction

Structural pattern recognition approach concerns with the description and classification of objects, taking into account the relations between their individual parts and in some extend ignoring the changes of geometry caused by different transformations. For example, two fingerprint images belonging to the same person change their appearance by the geometrical deformations occurring at the moment of capturing the impression. But, we can see that connections between ridges is maintained. In general, geometrical modifications are possible with, and without, changes in topology. But, it is not possible to change the topology without modifying the geometry. We can break the geometric object into cells of dimension 0,1,2,3, ... corresponding to vertexes, edges, faces, volumes... After

* Partially supported by the Austrian Science Fund under grants S9103-N13 and P18716-N13.

that, we can extract relevant topological information from the object. In digital imagery, the application of the algebraic topology tools has been started to use [1]. To do that, we may consider the set of pixels as a cubical cell partition of the image space. This is possible because homology of an image does not depend on its subdivision as long as each cell that composes the object is homeomorphic to a topological ball. Moreover, cubical homology theory has advantages, due to its ability to handle pixels or voxels directly, without an artificial triangulation of every pixel or voxel. Computing homology generators aims locating and characterizing the holes in a topological space. We may calculate n non-trivial homology groups $H_0, H_1, \cdots, H_{n-1}$ for a $n-$dimensional image. Informally, homology group H_p is a set of equivalence classes and every class is associated with a $p-$hole in this dimension. In pattern recognition, we want to know the number of $p-$ holes in the geometric object. But, we also want to control the geometry of the holes obtained. This aspect is usually avoided by the classical homology algorithms. The problem of efficient computation of homology groups and their ranks has been addressed from different points of view. As the classical homology algorithms [2]reduce the problem to Smith diagonalization, various optimizations strategies has been proposed [3–6]. However, this is not enough in many applications when the number of cubes or triangles is counted in thousands or more.

A second approach is the methods of reducing the numbers of cell of an object. Then, the homology groups of an object homologically equivalent, but composed of less cubes or simplices are computed. This kind of algorithms is efficient only if one step of the reduction is computationally inexpensive and the reduction in the number of cells is significant [7, 8].

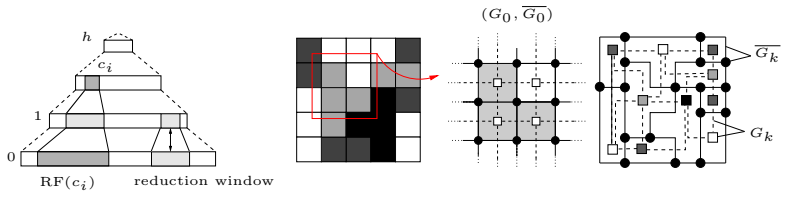
In this paper, our motivation is analyze the homology groups and generators that are obtained by the method proposed in [9] for 2D digital images. In this methods, there are the intention of controlling the geometry of the generator obtained and the authors claim that the generators computed seem to stay on boundaries. In general, we observe that a set of generators for a group H_1 is a linear combination of $1-$holes. Hence, we loss the control about the geometry of every particular hole. We want to visualize the generator at different levels of pyramid, and understand the influence of the contraction process about the generators obtained on the top level.

In Section 2, basic notions related to irregular graph pyramids, homology and the method to computing homology generators in a graph pyramid are recalled. Sections 3 and 4 present the proposed algorithms, followed by experimental results. Section 5 concludes the paper and gives an outlook of the future work.

2 Recall

2.1 Irregular Graph Pyramids

A *graph pyramid* P [10] is a stack of successively reduced graphs $P = \{G_1, \ldots, G_h\}$. Each level $G_k = (V_k, E_k)$, $0 \leq k \leq h$, is obtained by *contracting* and *removing*



a) Discrete levels b) Image to dual graphs

Fig. 1. a) Pyramid concept, and b) representation of the cells and their neighborhood relations by a pair of dual plane graphs at the level 0 and k of the pyramid.

edges in the level G_{k-1} below. Successive levels reduce the size of the data by a reduction factor $\lambda > 1$. Edges and vertices of the graphs G_k can be weighted.

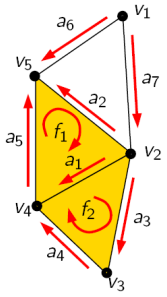
The *reduction window* relates a cell at the reduced level with a set of cells in the level directly below. The contents of a lower resolution (in a higher level) cell are computed by means of a *reduction function*, the input of which are the descriptions of the cells in the reduction window. Higher level descriptions should be related to the original input data in the base of the pyramid. This is done by the *receptive field* of a given cell $v \in G_k$. The receptive field of v aggregates all cells in the base level of which v is an ancestor. Each level represents a partition of the base level into connected subgraphs i.e. *connected subsets of pixels*, if the pyramid is build in the context of an image. The construction of an irregular pyramid is iteratively local [11]. On the base level (level 0) of an irregular pyramid the cells represent single pixels and the neighborhood of the cells is defined by the 4/6/8-connectivity of the pixels. A cell on level k + 1 (parent) is a union of neighboring cells in level k (children). This union is controlled by so called *contraction kernels* (CK) [12], a spanning forest which relates two successive levels of a pyramid. Every parent computes its values independently of other cells on the same level. Thus local independent (and parallel) processes propagate information up and down and laterally in the pyramid.

Higher level descriptions are related to the original input by the *equivalent contraction kernels* (ECK). 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 $\overline{G_k}$ (Fig. 1b). The vertices of G_k represent the cells on level k and the edges of G_k represent the neighborhood relations of the cells, depicted with square vertices and dashed edges in Fig. 1b. The edges of $\overline{G_k}$ represent the borders of the cells on level k, solid lines in Fig. 1b, including so called pseudo edges needed to represent neighborhood relations to a cell completely enclosed by another cell. Finally, the vertices of $\overline{G_k}$ (circles in Fig. 1b), represent junctions of border segments of $\overline{G_k}$. The sequence $(G_k, \overline{G_k})$, $0 \leq k \leq h$ is called irregular (dual) graph pyramid. For simplicity of the presentation the dual $\overline{G}$ is omitted afterward.

In [13], methods for optimally building irregular pyramids are presented. Methods like MIS and MIES ensure logarithmic height by choosing efficient contraction kernels i.e. contraction kernels achieving high reduction factors.

2.2 Homology

In this part, the basic notions of homology theory are recalled. Interested readers can find more details in algebraic topology classic books as [2] and others like [14–16].



$$E^1 = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{array} \begin{array}{c} \begin{array}{cc} f_1 & f_2 \end{array} \\ \begin{pmatrix} 1 & -1 \\ -1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \end{array}$$

Fig. 2. a): a simplicial complex made of 1 connected component and containing one 1-dimensional hole. b):Incident matrix describing the boundaries of each 2-cell f_1 and f_2

Starting from a cell decomposition of an object X its homology can be defined in an algebraic way by studying incidence relations of its subdivision. A cell of dimension p is called a $p-$cell. For example, in the simplicial complex illustrated on (Fig.2a) f_1 and f_2 are $2-$ cells; a_1,a_2,a_3,a_4,a_5,a_6 and a_7 are $1-$cells; v_1,v_2,v_3,v_4 and v_5 are $0-$cells. As it is shown on Fig.2 b: incidence matrix E^p describes the boundary of each $(p+1)-$cell.

The notion of $p-chain$ is defined as a sum $\sum_{i=1}^{nb\ p-cells} \alpha_i c_i$, where c_i are $p-$cells of X and α_i are coefficients assigned to each cell in the chain. For example, on (Fig.2a)the sums: $f_1 + f_2$ is a $2-$chain; $a_1 + a_4$, a_3 and $a_2 + a_7 + a_4$ are $1-$chains. Note that the notion of chain is purely algebraic and the cells that compose a chain do not have to satisfy any property of adjacency.

We compute the homology with coefficients over $\mathbb{Z}/2\mathbb{Z}$, because our image are nD objects embedded in $\mathbb{R}^D$. Note that in this case, a cell that appears twice on a chain vanishes, because $c + c = 0$ for any cell c when using moduli 2 coefficients (*i.e.* if a cell appears even times we discard it otherwise we keep it).

For each dimension $p = 0, \ldots, n$, where $n = dim(X)$, the set of $p-$chains together with a binary operation that define the sum of $p-$chains, forms an abelian group denoted C_p. Homology examines the connectivity between two immediate dimensions. To do so, and since for each dimension p we have a group, we may define a set of maps ∂_p between them to relate their structure. These applications ∂_p describe the boundary of $p-$chain as $(p-1)$chains. They are homomorphism and preserve the identity, inverses, and subgroups between group [16].

The $p-$chain groups can be put into a sequence, in the following way:

$$C_n \xrightarrow{\partial_n} C_{n-1} \xrightarrow{\partial_{n-1}} \cdots \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} 0, \tag{1}$$

which satisfy $\partial_p \partial_{p-1}(c) = 0$ for any $p-$chain c. This sequence of groups is a *free chain complex*.

The boundary of a single p-cell is defined as the sum of its incident $(p-1)-$cells. As these applications are lineal by definition, the boundary of a general $p-$chain is then defined by linearity as the sum of the boundaries of each cell that appears in the chain e.g. in (Fig.2a) $\partial(f_1 + f_2) = \partial(f_1) + \partial(f_2) = (a_1 + a_2 + a_5) + (a_1 + a_3 + a_4) = a_2 + a_3 + a_4 + a_5$. Note that chains are considered over $\mathbb{Z}/2\mathbb{Z}$ coefficients i.e. any cell that appears twice vanishes. For each dimension p, the set of p-chains which have a null boundary are called *p-cycles*. The null boundary is the element identity for the defined binary operation on group C_p. Hence, the $p-$cycles conforms the kernel of ∂_p, which is a special subgroup in their domain C_p, denoted Z_p. e.g. $a_1 + a_3 + a_4$ and $a_1 + a_5 + a_6 + a_7$ are $1-$cycles. The set of p-chains which bound a $p+1$-chain are called *p-boundaries* and they are a subgroup of C_p, denoted B_p e.g. $a_1 + a_2 + a_5 = \partial(f_1)$ and $a_2 + a_3 + a_4 + a_5 = \partial(f_1 + f_2)$ are $1-$boundaries. As we have seen the boundary of a boundary is the null chain. This imply that all boundaries are cycles and it is possible to see that B_p is a subgroup of Z_p. Note that every $0-$chain is a cycle. Recall that we are interested in characterize the 'holes' and a $p-$hole is a $p-$cycle which is not a $p-$boundary. e.g z$=a_2 + a_6 + a_7$ is a $1-$dimensional hole. We may factor Z_p using the subgroup B_p, getting the cosets. The elements in the cosets of B_p correspond to $p-$cycles which are not $p-$boundaries. For $z \in Zp$, the subset $z + B_p = \{z + b | b \in B_p\}$ of Z_p is the left coset of B_p containing z. The element z is called its representative. If two elements z_1 and z_2 of Z_p are representatives for the same coset then $z_1 + B_p = z_2 + B_p$. It is possible to demonstrate that $b = z_1 + z_2 \in B_p$. Hence, two $p-$cycles z_1 and z_2 are in the same coset iff there exist a boundary $b \in Bp$ with $z_1 = z_2 + b$. (Note that with coefficients in $\mathbb{Z}/2\mathbb{Z}$, the inverse of z is z).

This defines an equivalence relation (homology relation) in the group of $p-$cycles. The set of $p-$cycles Z_p is then partitioned by the homology relation, according to the hole they surround. Two $p-$cycles in the same equivalence class said to be homologous. In conclusion, The p^{th} homology group, denoted H_p, is defined as the quotient group Z_p/B_p. Thus, elements of the homology groups H_p are equivalence classes and two cycles z_1 and z_2 belong to the same equivalence class if their difference is a boundary. For example, in figure Fig.2 a: $B_1 = \{a_1 + a_3 + a_4, a_1 + a_2 + a_5, a_2 + a_3 + a_4 + a_5\}$ is the set of 1-boundaries, and $Z_1 = \{a_1 + a_3 + a_4, a_1 + a_2 + a_5, a_2 + a_3 + a_4 + a_5, a_2 + a_6 + a_7, a_1 + a_5 + a_6 + a_7, a_3 + a_4 + a_5 + a_6 + a_7, a_1 + a_2 + a_3 + a_4 + a_6 + a_7\}$ is the set of $1-$ cycles. Hence, $H_1 = Z_1/B_1 = \{\{a_2 + a_6 + a_7, a_1 + a_5 + a_6 + a_7, a_1 + a_2 + a_3 + a_4 + a_6 + a_7, a_3 + a_4 + a_5 + a_6 + a_7\}\}$. We can see that: $a_2 + a_6 + a_7 = a_1 + a_5 + a_6 + a_7 + \partial(f_1)$; $a_2 + a_6 + a_7 = a_1 + a_2 + a_3 + a_4 + a_6 + a_7 + \partial(f_2)$; $a_2 + a_6 + a_7 = a_3 + a_4 + a_5 + a_6 + a_7 + \partial(f_1 + f_2)$; $a_1 + a_5 + a_6 + a_7 = a_1 + a_2 + a_3 + a_4 + a_6 + a_7 + \partial(f_1 + f_2)$; $a_1 + a_5 + a_6 + a_7 = a_3 + a_4 + a_5 + a_6 + a_7 + \partial(f_2)$; and $a_1 + a_2 + a_3 + a_4 + a_6 + a_7 = a_3 + a_4 + a_5 + a_6 + a_7 + \partial(f_1)$.

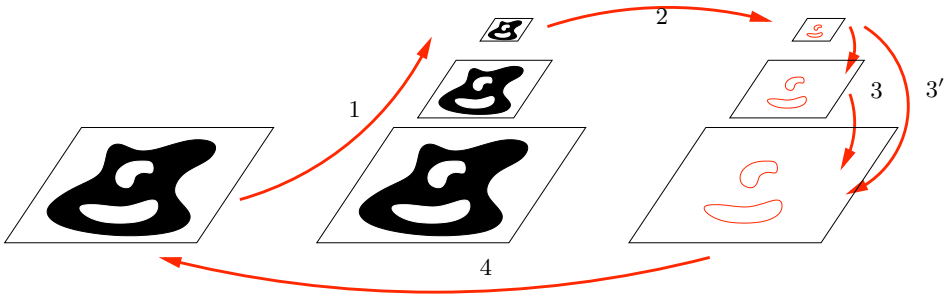


Fig. 3. Illustration of the methods use in [9] for computing generators of homology groups using an image pyramid.

A set of generators for a group H_p is defined as a maximal subset S of elements of H_p, such that every element of H_p can uniquely be defined as a linear combination of element of S [17].

2.3 Computing Homology Generators in a Graph Pyramid

The method in [18] follows the approach of reducing the number of cells of an object in order to compute homology. This has a similar idea as used in [7, 17]. But, in this case all simplifications that are computed during the reduction process are kept by using a pyramid as is illustrated in (Fig. 3). The approach builds a hierarchical structure using two operations: contraction and removal. In this way, homology generators were computed in the top level of the pyramid, and could be used to deduce generators of any level of the pyramid. The method can be summarized in three steps:

1. Starting from a segmented image, built a graph pyramid using contraction kernels of cells with the same label.
2. Homology generators are computed in the top level of the pyramid.
3. Deduce the homology generators of lowest level directly from the highest level using the notion of equivalent contraction kernel.

The visualization of homology generators that were computed on top level of pyramid was only possible by down-projecting to the base level and showing the image itself. So far, was not possible to visualize them in the top level neither the projections on the next intermediate levels.

3 Visualizing a Graph Pyramid

In this section, a novel algorithm to visualize connecting paths of dually contracted graphs is shown. In the process of building a pyramid, the number of self loops and multiple edges that produces the contractions are increasing, and most

of them can not be eliminated in the simplification process, see Fig. 4 c). A very simple method to draw such graphs using straight lines [19], can not deal with this problem satisfactorily making difficult the study of particular situations in upper levels of pyramids. We propose a new method to nicely draw multiple edges and self loops maintaining the correspondences between surviving nodes.

The general problem is to draw a plane (multi)graph $G_k = (V_k, E_k)$ such that edges do not cross, assuming that the planar graph $G_0 = (V_0, E_0)$ is defined on a square grid and G_k has been constructed by dual graph contraction. The nodes has fixed positions in the plane and they should be kept in the whole drawing of pyramid. The general steps of the new algorithm are enumerated as follows:

1. Determine the Equivalent Contraction Kernel (ECK) that corresponds to graph G_k, i.e. $(S_k, N_{0,k})$.
2. For all the edges $e \in E_k$, find the corresponding bridge in E_0.
3. Calculate multiplicity of ECK-branches, i.e. count the number of bridges the branch is connected to.
4. Place interconnection squares:
   - where branches split, or
   - where they change direction
5. Draw parallel lines according to the multiplicity count between all connected interconnection squares.
6. Connect lines inside interconnection squares without crossings.

The first step is to obtain the ECK that leads to graph G_k, meaning the set of edges contracted from the base level graph that produces the actual graph. In the Fig. 4 a), it is shown the base level graph example and in b) the EKC that produces the graph in c). In general, this can be seen recursively as the ECK of the level k-2 plus the bridges of the edges in the CK of level k-1 ($N_{0,2} = N_{0,1} \cup$ BRIDGES($N_{1,2}$)). The second step is to find the corresponding bridges of all the edges from graph G_k, these edges plus the ones from the ECK obtained in the level before will be used as possible paths to draw connecting edges between surviving nodes. Note that each corresponding bridge from edges of graph G_k is connecting two contracted trees, making the actual drawing connected. Dual graph contraction ensures the existence of connecting paths and that each connecting path contains exactly one bridge. In case of multiple equivalent connecting paths, that are not yet eliminated by the simplification process, one can be selected arbitrarily or as the path which is part of the ECK of the apex. The third step should start by the leafs of $N_{0,k}$ with multiplicity initialized in 0 and incremented by one for all incident bridges of step 2. All "interior" edges of $N_{0,k}$ sum the multiplicity of their sons plus, eventually, the number of incident bridges, see Fig. 4 d).

The set of edges that has multiplicity 0 are not useful to the drawing so they are eliminated. Taking into account only the rest of the edges, for each "interior" node (non surviving node) we check if there exist a branches split or direction change to place the interconnection squares. In the case of surviving nodes an interconnection square is placed only if contains an adjacent edge with order greater than 1, see Fig. 4 e).

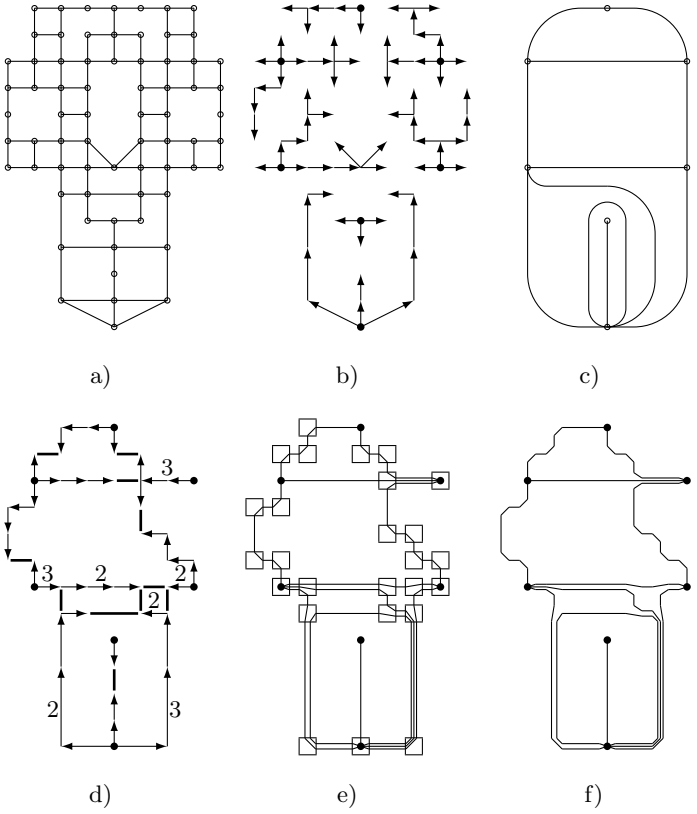


Fig. 4. Process of drawing the graph of level 2 (c) from example pyramid with base level (a).

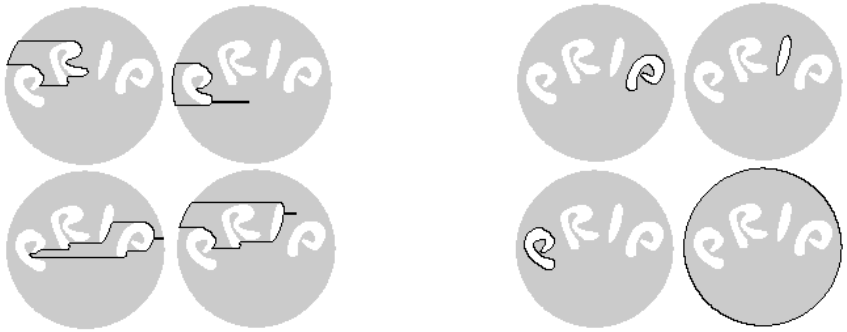
The interconnection squares where the edge orientation changes without splitting of branches have one side through which all lines enter and one side through which all lines exit. Hence, their order can be preserved without crossing. The interconnection squares at a branching correspond to a branching of a contraction tree (e.g a connected component of the ECK). Consequently there is only of the square through which the lines reach the tree's root. The multiplicity of this side is the sum of the multiplicity of all the other (3) sides. Furthermore, there is a strict order (i.e. clockwise) of the remaining three sides and the line bundles can be connected in the same order as shown, see Fig. 4 e). Finally, interconnection squares are not shown in the drawing, see Fig. 4 f).

4 Experiments

Until now, the visualization of homology generators, computed in a graph pyramid, was possible only by down-projecting them and then visualizing them in

the base level (image). Also, the homology generators are computed in the dual graphs $\overline{G_k}$, but the visualization was done in the image itself (see Figure 5).

With the new method we are able to see all levels of pyramid and in particular the top levels where generators are computed (Fig. 6). Now, we are also able to see the generators projections on the following levels, see Fig. 7. The figure shows the experiments using the images from [18]. The image a-) has 3 generators represented by numbers, the generator number 1 and 2 are self loops, and the third one is divided in two edges. The projection of this generators in the previous level(d), gives the generator 1 as a self loop but the generator 2 is now divided in two edges and the generator 3 is divided in 3 edges as well. The top level contains exactly a set of generators of the initial image. As shown, the graphs in those levels have self loops that now are nicely drawn and we can easily see the set of generators obtained by the method. The new method can correctly visualize all pyramid levels and any generators computed on them.



(a) (b)

Fig. 5. Homology generators are computed in dual graphs but visualized on original image, (a): without using pyramids, (b): using pyramids and down-projecting them.

5 Conclusion

This paper presents a method for nicely visualize graph pyramids with multiple edges and self loops using not just straight lines, preserving the geometry and topology of the original image. The usefulness of the method is shown, but not limited to, the context of homology generator computation using irregular graph pyramids. We plan to extend this method for 3D combinatorial map pyramids, and apply it in the context of homology generator computation in 3D.

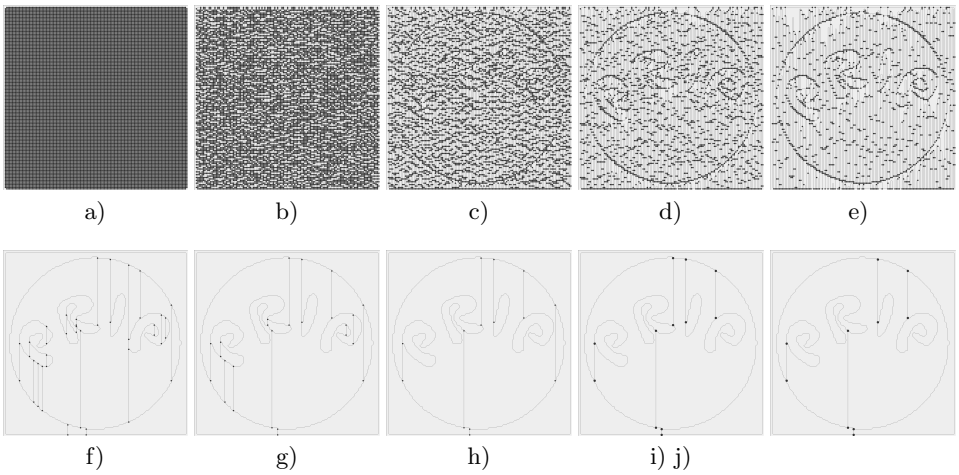


Fig. 6. Graph Pyramid drawing. First raw shows the first 0-4 levels and second raw last 30-34 levels.

References

1. R.Gonzalez, P.Real: On the cohomology of 3d digital images. Discrete Appl. Math. **147**(2-3) (2005) 245–263
2. Munkres, J.R.: Elements of algebraic topology. Perseus Books (1984)
3. R.Kannan, A.Bachem: Polynomial algorithms for computing the smith and hermite normal forms of an integer matrix. SIAM J. Comput. (8) (1979) 499–507
4. C.J.A.Delfinado, H.Edelsbrunner: An incremental algorithm for betti numbers of simplicial complexes on the 3-spheres. Comput. Aided Geom. Des. **12**(7) (1995) 771–784
5. G.Carlsson, V.de Silva: A geometric framework for sparse matrix problems. (33) (2004) 1–25
6. A.Zomorodian, G.Carlsson: Computing persistent homology. SCG '04: Proceedings of the twentieth annual symposium on Computational geometry (2004) 347–356
7. Kaczynski, T., Mrozek, M., Slusarek, M.: Homology computation by reduction of chain complexes. Computers and Math. Appl. (34) (1998) 59–70
8. W.D.Kalies, K.Mischaikow, Watson, G.: Cubical approximation and computation of homology. Banach Center Publ. (47) (1999) 115–131
9. Peltier, S., A.Ion, Haxhimusa, Y., Kropatsch, W., Damiand, G.: Computing of homology group generators of images using irregular graph pyramids. GBRPR 2007, LNCS **4538** (2007) 283–294
10. Jolion, J.M., Rosenfeld, A.: A Pyramid Framework for Early Vision. Kluwer (1994)
11. Meer, P.: Stochastic image pyramids. Computer Vision, Graphics, and Image Processing **45**(3) (March 1989) 269–294 Also as UM CS TR-1871, June, 1987.
12. Kropatsch, W.G.: Building irregular pyramids by dual graph contraction. IEE-Proc. Vision, Image and Signal Processing **142**(6) (December 1995) 366–374
13. Kropatsch, W.G., Haxhimusa, Y., Pizlo, Z., Langs, G.: Vision pyramids that do not grow too high. Pattern Recognition Letters **26**(3) (2005) 319–337

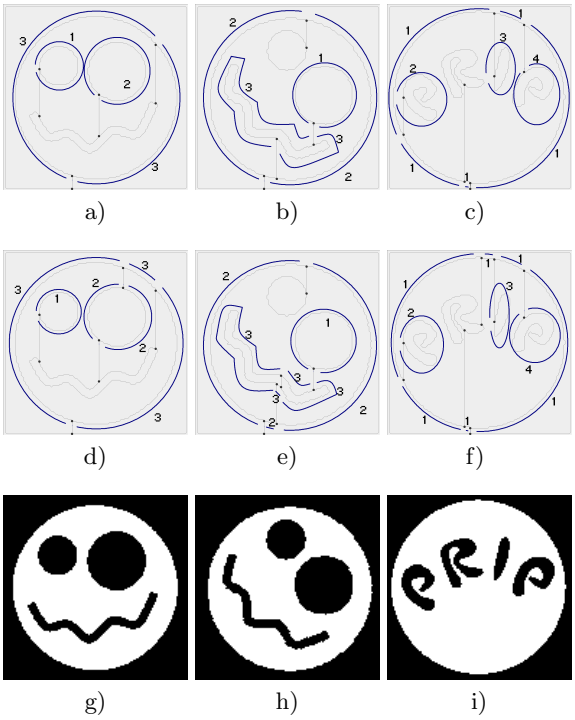


Fig. 7. First two top levels of pyramids showing generators. The first raw shows the generators in the top levels and the second raw the projected generators into the previous level. The third raw shows the original images.

14. Hajime, S.: Algebraic Topology: An Intuitive Approach. American Mathematical Society (1999)
15. Kaczynski, T., K.Mischaikow, M.Mrozek: Computational Homology. Springer Verlag (2004)
16. A.J.Zomorodian: Topology for computing. Cambridge University Press (2005)
17. G.Damiand, S.Peltier, L.Fuchs: Computing homology for surfaces with genralizaed maps: Application to 3d images. ISVC 2006, LNCS, **4292** (1998) 235–244
18. Peltier, S., Ion, A., Haxhimusa, Y., Kropatsch, W.G., Damiand, G.: Computing homology group generators of images using irregular graph pyramids. In Escolano, F., Vento, M., eds.: Proceedings of the 15th International Workshop on Graph-based Representation for Pattern Recognition. Volume 4538 of Lecture Notes in Computer Science., Alicante, Spain, Springer, Berlin Heidelberg, New York (June 2007) 283–294
19. Kerren, A., Breier, F., Kugler, P.: Dgcvis: An exploratory 3d visualization of graph pyramids. In: CMV '04: Proceedings of the Second International Conference on Coordinated & Multiple Views in Exploratory Visualization, Washington, DC, USA, IEEE Computer Society (2004) 73–83