20. Vision by Graph Pyramids

Walter G. Kropatsch*

Abstract

To efficiently process huge amounts of structured sensory data for vision, graph pyramids are proposed. Hierarchies of graphs can be generated by dual graph contraction. The goal is to reduce the data structure by a constant reduction factor while preserving certain image properties, like connectivity. While implemented versions solve several technical vision problems like image segmentation, the framework can be used as a model for biological systems, too.

*I would like to thank Yll IIaxhimusa for the improved illustrations on building an example graph pyramid. I would like to thank the reviewers and editors of this collection for their helpful comments to enhance the general readability of this chapter. This work was supported by the Austrian Science Foundation under grants S 7002-MAT, P 14662-INF and P 14445-MAT.

I. Introduction

Animals and mechanical systems are equipped with many sensors. The individual sensor elements are spatially distributed in many different arrangements. While regular *sensor arrangements* like square grids dominate the fabricated world, most of their biological counterparts show a regular distribution only on a large scale, their sensor neighborhoods are neither geometrically nor topologically regular. Figure 1 shows portions of the sensor arrangements of a typical digital camera and a monkey's retina. The sensors' positions are indicated by a small circle, the number of elements in Fig. 1 has been chosen to match approximately the 2339 sensory elements of the monkey retina. The processing of the data measured by the sensors involves a huge number of processing elements (e.g. processors, neurons) that are interconnected in very complex ways. In this paper we propose one possible interconnection network, the *graph pyramid*, that combines two data structures in an efficient way: logarithmic pyramids (see Jolion and Rosenfeld 1994 for a survey) and attributed relational graphs (for theory see Thulasiraman and Swamy 1992, for recent applications in Pattern Recognition see Jolion and Kropatsch 1998, Kropatsch and Jolion 1999, Jolion et al. 2001). Regular pyramids are built by repeatedly reducing the resolution (and the size) of the

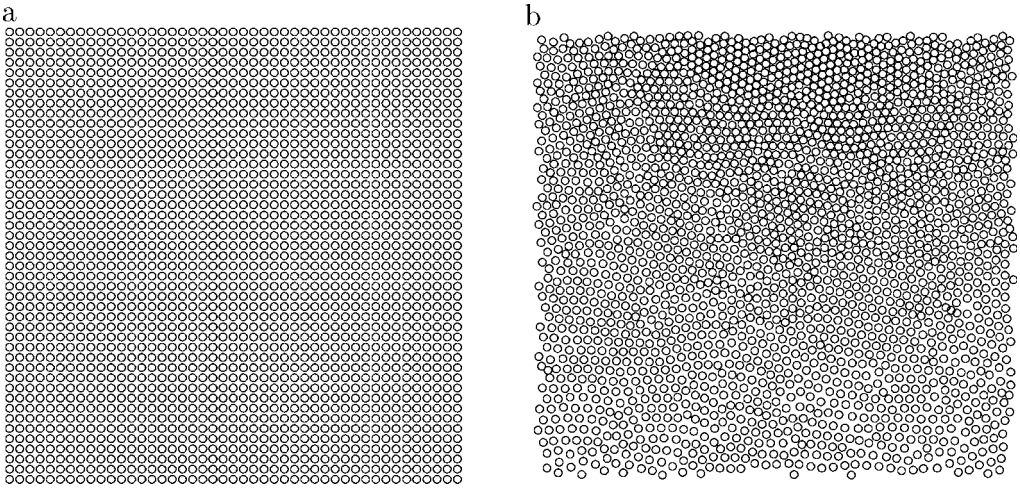

Fig. 1. Sensor arrangements, a) part of CCD chip, b) part of monkey's retina (data kindly provided by P Ahnelt)

image by a constant factor, e.g. a $1024 \times 1024 = 2^{10} \times 2^{10}$ image produces a pyramid with 10 levels above the base: 512×512, 256×256, 128×128, 64×64, 32×32, 16×16, 8×8, 4×4, 2×2, 1×1. The reduced images are computed by local (weighted) averages relating the (gray) values of the reduced pyramid level with a small number of pixels in the level below. These 'vertical' connections between the images in the pyramid stack establish short (logarithmic) connections between the apex of the pyramid and any pixel in the base image. The resulting computational efficiency has been used in many vision systems to speed up their performance. But pyramids are also used as a model to explain visual perception in biological systems (e.g. see Uhr and Schmitt 1984, Pizlo et al. 1997).

Sensors typically measure a single quantity like the light intensity butobjects in the environment are connected volumetric entities, they have a 'body', a certain shape and can be *decomposed into parts*. For example, a human face is characterized by its eyes, its nose and its mouth, and, what is considered a major structural property, the mutual relations between its parts.

Let us now study the first stages of visual information processing and focus on the data representation needed at the different levels of processing. Through the projection of scene objects of the environment into the image, object patches with same color or texture are mapped into image regions of same characteristics. By the discrete sampling of these objects, the corresponding regions are decomposed into many similar sensor signals, adjacent sensors may sense the same color, similar light intensity or even complicated texture features. 'Image Segmentation' finds homogeneous regions in the sampled image and determines the spatial relations between these regions. Region adjacency allows a system to group all region patches of one object or to find groups of objects that belong together.

A simple and efficient way to represent all three types of structures: the arrangements of sensor elements, the region adjacencies, and the part-whole relations of objects, is an *attributed relational graph* (ARG). It consists of a set of nodes or

vertices V, of a set of edges E with each edge relating two vertices. Both vertices and edges may receive numerical or symbolic attribute values to specify particular properties. A pixel array can be converted to an ARG without loss of information by simply defining the pixels as the vertices of the graph with a gray value attribute. Neighboring pixels create an edge between the vertices corresponding to the pixels. But the ARG goes beyond the representational capabilities of the pixel array. Vertices can be identified with regions or also with objects. In these cases an edge can express region adjacency or mutual object relations respectively. In addition, the ARGs that are derived from images are embedded in the image plane: such graphs are called 'plane graphs'[1].

Since the sensor elements form a surface (like the eye's retina), we represent their topological arrangement by a pair of graphs, which are non-simple, plane, and dual to each other. (Large) connected regions and their topological relationships often allow the inference of spatial relations in the environment and are very robust with respect to noise and sensing inaccuracies. These properties are preserved by dually contracting the graphs, operations which can be implemented in a massive parallel and local system architecture. Repeated contraction yields a stack of successively reduced graphs: *a graph pyramid*.

This paper is organized as follows: We first examine the different processing stages of image analysis by a computer system (Section III). We then study the purpose of vision both for computers and for biological systems (Section IV). Section V explains the basic concepts to build a graph pyramid. This notion of "*equivalent contraction kernel*" allows us in section V.C to define all possible graph pyramids that can be built on top of a given base graph, e.g. the sensor arrangement of a retina. In the conclusion we refer to the different successful applications of pyramids and graph pyramids.

[1]Plane graphs should not be confused with planar graphs for which such embedding exists. E.g. a different choice of the background face may cause a different embedding in the plane.

II. Processing Stages of Computer Vision

Let us consider vision as a process that is supposed to capture the essential structure of the world. The cyclic arrangement of the diagram in Fig. 2 shows the structure of the environment of a 'seeing system' ('REALITY') and the internal stages of recovery of this structure from the image. It tries to relate the structure of the WORLD on the left side with the reconstruction stages of the COMPUTER vision process on the right side. A DIGITAL IMAGE, as captured by a *sensing camera*, is an array of measurements called pixels (picture elements). Simple *image processing* consists in modifying the individual pixel values in order to stretch the image contrast, to remove noise, and to compute features used in segmentation to classify pixels as belonging to a specific image object or to the background. *Segmentation* collects all pixels of the same class into connected sets of pixels: the image REGIONS. These regions create *region adjacency* relations between the regions. Geometric region properties like "shape" allow the *identification* of SPECIFIC PARTS corresponding to the REFLECTING SURFACES of the reality. Using knowledge about the structure of objects, the specific parts can be further *assembled* into image OBJECTS. Together with their mutual relationships they constitute the SCENE DESCRIPTION describing (and interpreting) the content of the visual input.

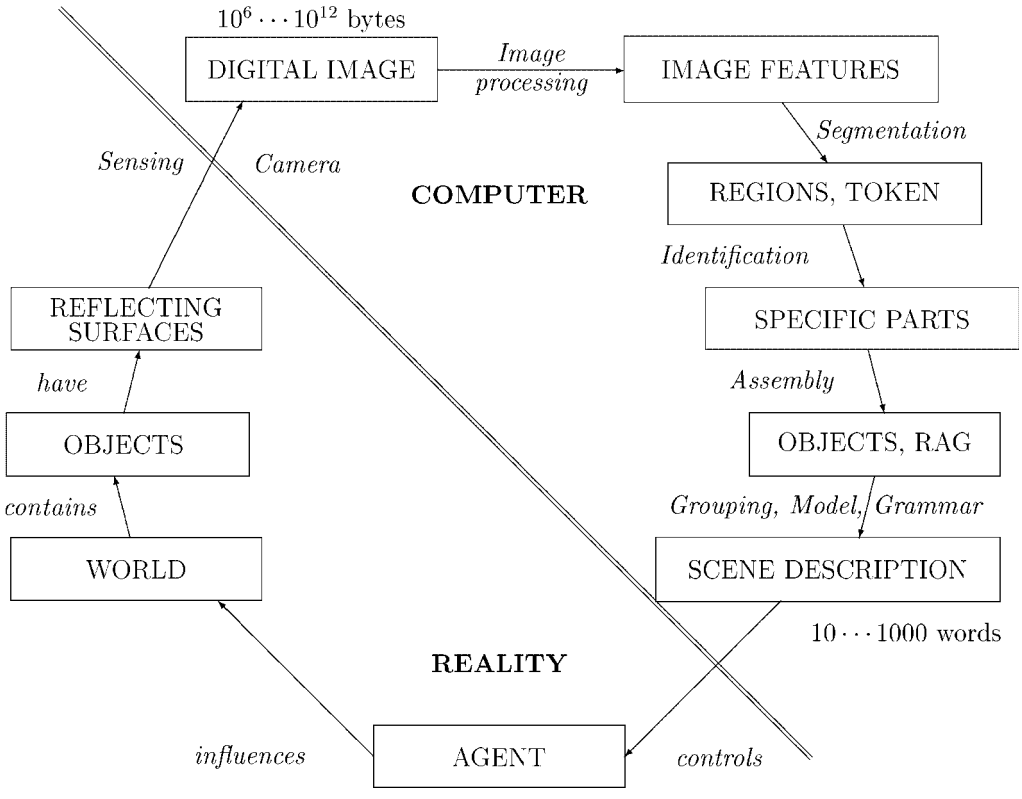

Fig. 2. The vision process as a cycle

Figure 3 illustrates some of the vision steps by means of a very simple example. 11 objects, black squares on white background, are sampled in a square grid producing 196 pixels of which 44 fall on the black object surface. Pairs of pixels are considered adjacent if they are neighbors in the same row or the same column (4-adjacency). The corresponding neighborhood graph contains 364 Edges. Segmentation and connected component analysis[2] assigns each group of 4 black pixels a unique label, $1, 2, \ldots 11$, each corresponding to one object. The white background forms a single connected component having 11 holes (e.g. the 11 black squares). Since the fact that all the 11 objects are adjacent to the background is not the ultimate result (i.e. also the characters of a scanned document), spatial closeness is used to describe their arrangement in the image. The resulting object graph consists of 11 object-vertices and 18 object-relations. The later reflects the placement of the objects in the image plane[3] by relating objects which are closest in the image. Such graphs are compact and very convenient data

[2]A connected component is a set of pixels which have the same gray value and which are connected.

[3]Geometric object properties like coordinates or size are kept in numerical attributes of the vertices.

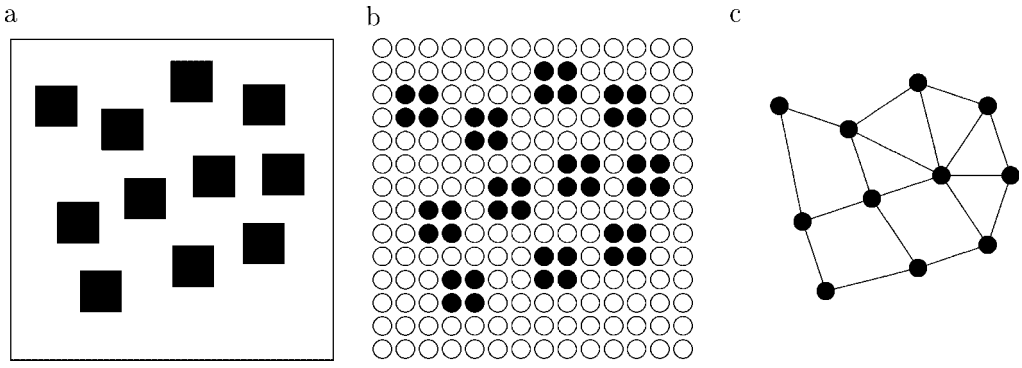

Fig. 3. Deriving the structure of 11 objects after regular sampling. *a* WORLD, *b* binary DIGITAL IMAGE, *c* SCENE DESCRIPTION

structures to describe topological relations among image entities at multiple levels in a consistent way.

III. Purpose: Smart and Efficient Decisions

Before studying the sensors and sensing processes that extract information from the environment let us ask for the purpose of these complicated and not yet sufficiently understood systems. Pizlo et al. (1997) define the goal of visual perceptions as follows: '*The goal of vision perception is to provide the observer with visual information about the 3D environment so that the observer can recognize objects*, *manipulate them*, *and navigate in the environment*.' A natural environment typically contains an enormous variety and amount of objects, even a single snap shot could not be fully interpreted in reasonable time. Hence the need to efficiently focus on those pieces of information that are *relevant* to take appropriate decisions. Less important data can be neglected, but their removal should not disturb the (probably) important spatial relations between the relevant parts. This brings us back to the formulation of Marr (1982): '*building a description of the shapes and positions of things from images*'. We conclude that vision algorithms must be capable of *simplifying a huge amount* of sensory data without loosing the relevant information. Since sensors have different tasks the purpose varies and the respective algorithms must allow conscious control and adaptation to the actual purpose in a smart way. In this context, *adaptation* means that other weights of importance are given to the features derived from the data depending on the actual purpose. A different goal may need a different decision to be drawn from the same visual data.

IV. The Graph Pyramid

Connected objects are mapped into connected image regions if the image's projection is large enough to satisfy the discrete sampling theorem (e.g. to inscribe a circle with a radius larger than the sampling distance). Geometrical measurements derived from a digital image are sensitive to errors due to noise, discrete sampling and motion inaccuracies. However, the structural and topological relations like region-adjacency or part-whole are inherent to the objects and their

arrangement in the image up to discretization. In many cases, they do not depend on the particular imaging situation. This is the background of several recent contributions describing spatial/structural representations and transformations preserving topological relations existing in the image plane. Let us enumerate a few approaches preserving structural relations:

1. The simplest most frequent representation uses coordinates as vertex attributes of an ARG. This immediate representation depends on the particular mapping geometry. For well controlled environments (e.g. geographic information systems) it is widely used due to its simplicity.
2. Rosenfeld and Nakamura (1997) consider local deformations of (digital) curves in the plane that preserve an implicitly given topology. The idea is that images showing the same topological arrangement of regions and curves can be transformed into each other continuously.
3. A pair of plane dual graphs is the base of a graph pyramid built by repeated dual graph contractions (Kropatsch 1997a). It differs from the previous approach in that the transformed data are reduced at each step by a constant reduction factor which is the origin of its computational efficiency.
4. In topological and combinatorial maps (Gareth and Singerman 1978, Lienhardt 1989) the embedding is determined by the local orientation of the structural elements. We have shown in Kropatsch and Brun (2000) how to perform dual graph contraction with combinatorial maps.

To associate a discrete image region with the corresponding surface patch of the object and to hypothesize adjacency of patches from adjacency of regions, the connectivity of the graph (or the subgraph representing the object) is an essential structural property. Since our goal is to successively remove unnecessary parts the connectivity can be lost by these operations. Before disconnecting a graph into two components these two components will be connected by a single edge which is called a *bridge* (Fig. 4). Hence bridges should not be removed, or, what is equivalent, the dual counterpart of bridges, self-loops, should not be contracted. Since our graphs are embedded in the image plane, each edge bounds two regions of the plane which are connected by the dual edge in the dual graph. The two regions bounded by a bridge are the same (R_0 in Fig. 4) and consequently the two extremities of the dual edge too: such an edge is called a self-loop.

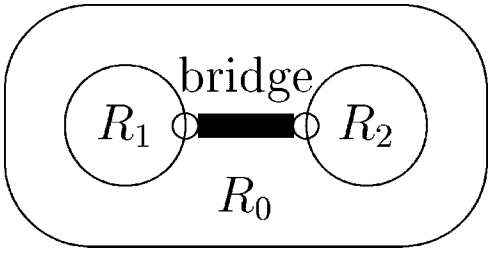

Fig. 4. The removal of the bridge would disconnect subgraphs R_1 and R_2

A. Dual Graph Contraction

Let us consider an example and discuss the dual contraction of a graph by means of this example. The complete formalism will follow.

1. Example: Connected Components of an Image

Consider a simple example (Fig. 5): Pixel gray values become the attributes of the corresponding vertices of the base graph

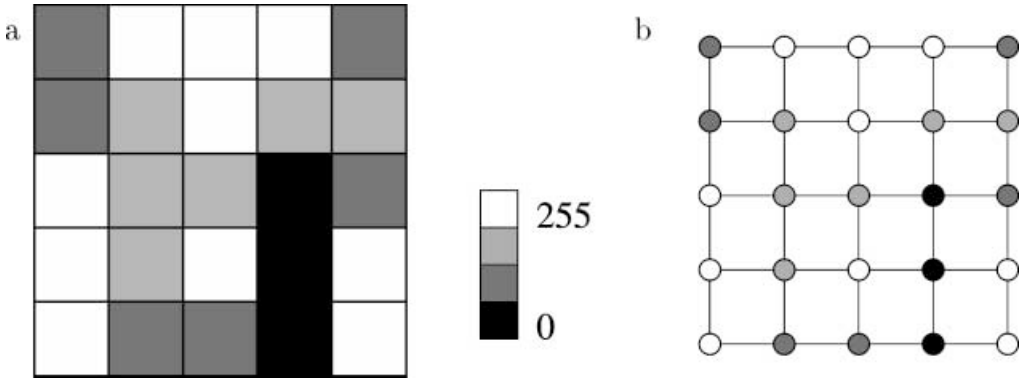

Fig. 5. Image to ARG conversion: *a* a 5×5 gray level image, *b* the corresponding base graph G_0

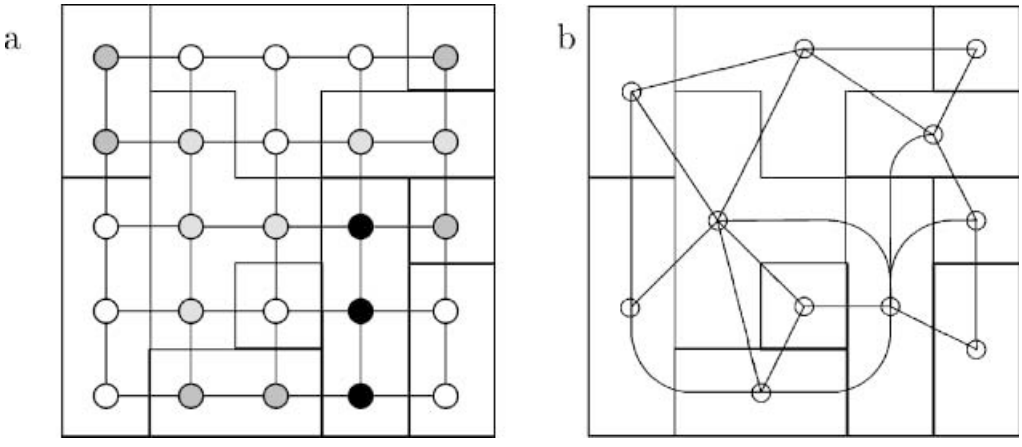

Fig. 6. To derive: the RAG of the connected components. *a* The 11 connected components of G_0. *b* The region adjacency graph (RAG) G_2

G_0 and are illustrated by the color with which the circle corresponding to the vertices are filled (Fig. 5b). Two vertices are connected by an edge (a line segment in Fig. 5b) if the two corresponding vertices share a boundary segment. The four different gray values/colors of the 25 pixels form 11 groups of adjacent pixels all having the same gray value: they are called the connected components (Fig. 6). Our processing goal is to represent each such group by one vertex and to connect two vertices if the corresponding pixel groups share at least one boundary segment (see Fig. 6b).

In order to derive the smaller graph G_2 from G_0 we apply several contraction operations to the graph until the final result is reached. The primitive operation of dual graph contraction is the contraction of an edge $e = (v, w)$. It consists of the identification of the two end vertices and the removal of the edge. We can choose v to 'survive' and substitute all appearances

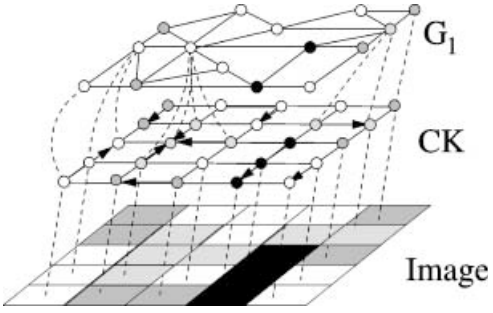

Fig. 7. Contraction kernels (CK) at the base level G_0 and the contracted graph G_1

of w in any of the edges by v. One can visualize the process dynamically by moving w along e into v while stretching all the edges attached to w.

If we select only edges for contraction the end points of which have the same gray value, the connected components will shrink but all the connections between different gray values will be preserved. In Fig. 7 the selected edges are marked by an arrow pointing towards the surviving vertex. All the edges that contract into the same surviving vertex are called 'contraction kernel'. If these kernels do not form cycles (e.g. each is a small tree) all contractions can be executed simultaneously.

Some of the surviving vertices may become multiply connected by multi-edges. In most cases only one of the multi-edges is needed (Fig. 8). But in some cases (Fig. 9) the multi-edge represents relevant information: when the two regions meet along two or more distinct boundary segments. In all previous cases an edge between two surviving vertices was associated with one connected boundary segment between the two regions. If we would remove one of the two edges in Fig. 9c this useful topological property would be lost. If we continue contracting one of the double edges we end up with a self-loop $e = (v, v)$ where both end points are identical. As we can see intuitively in Fig. 9d its removal would again destroy the above property. In addition we would loose the fact expressed by the self-loop that the inner region is completely surrounded by the other region. We conclude that both multiple edges and self-loops

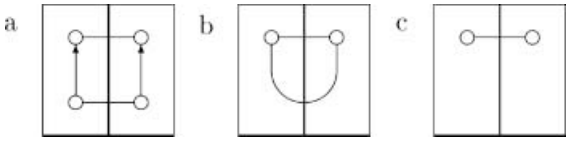

Fig. 8. Redundant double edge: *a* CK, *b* contracted, *c* simplified

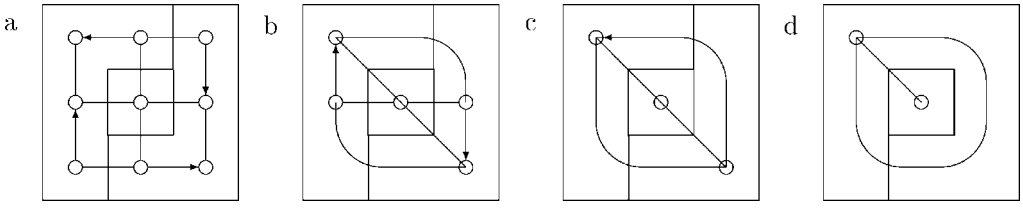

Fig. 9. Non-redundant double edge and self-loop: *a* $G_0 + CK_{01}$, *b* $G_1 + CK_{12}$, *c* $G_2 + CK_{23}$, *d* G_3 with self-loop

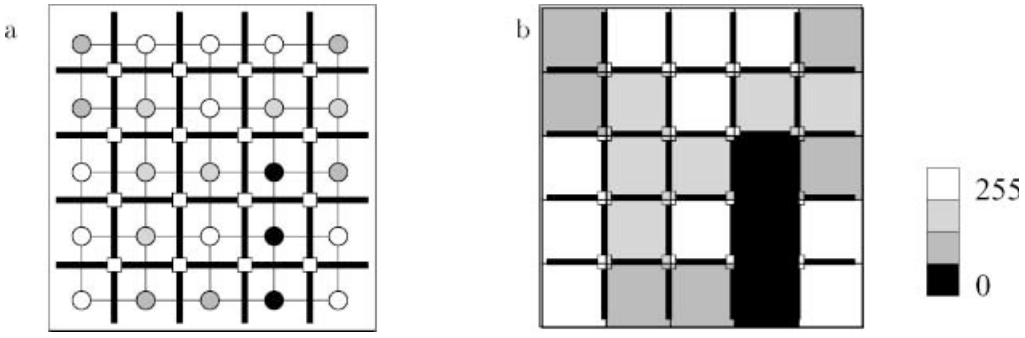

Fig. 10. The role of the dual graph $\overline{G_0}$: *a* the dual graphs (G_0, $\overline{G_0}$), *b* the dual graph $\overline{G_0}$ on the image

may be necessary to preserve the topological properties of the surviving components of the graph.

But how do we decide whether an edge is redundant or not? For this purpose we consider the dual graph. It can be constructed by placing a new (dual) vertex (depicted by a small square in Fig. 10) into each face of the drawn graph and connecting two new vertices across the edge of the primal graph which separates the two faces. For the pixel array this graph consists of the centers of all 2×2 blocks and the boundary segments separating two pixels (Fig. 10b). The contraction of an edge of the primal graph merges the two pixels corresponding to the edge's end points. Consequently the separating boundary segment e.g. the edge of the dual graph has to be removed. Edge contraction corresponds to edge removal in the dual graph. This allows to maintain duality between the dual graphs after 'dual graph contraction'. With the observation that the number of boundary segments of a face corresponds to the degree of the dual vertex let us reconsider the cases of multi-edge and of self-loop: the degree of the face bounded by the double edge in Fig. 8b is obviously two, the two faces between the double edges in Fig. 9c have both degree three. A similar observation holds for the self-loop: the face inside a removable self-loop has degree one, non-removable self-loops are characterized by an inside face of degree three (Fig. 9d) or higher.

A close look at the result of our first contraction (Fig. 7) shows that not all connected components have been contracted into a single vertex. We therefore repeat the selection of contraction kernels on graph G_1 and find four more edges to contract (Fig. 11). The resulting pyramid has three levels G_0, G_1, G_2 and yields at the apex the RAG of the original image. The dashed vertical lines in Figs. 7 and 11 indicate the correspondences between

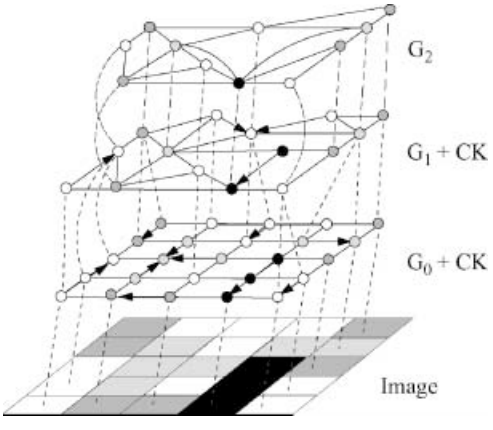

Fig. 11. The pyramid

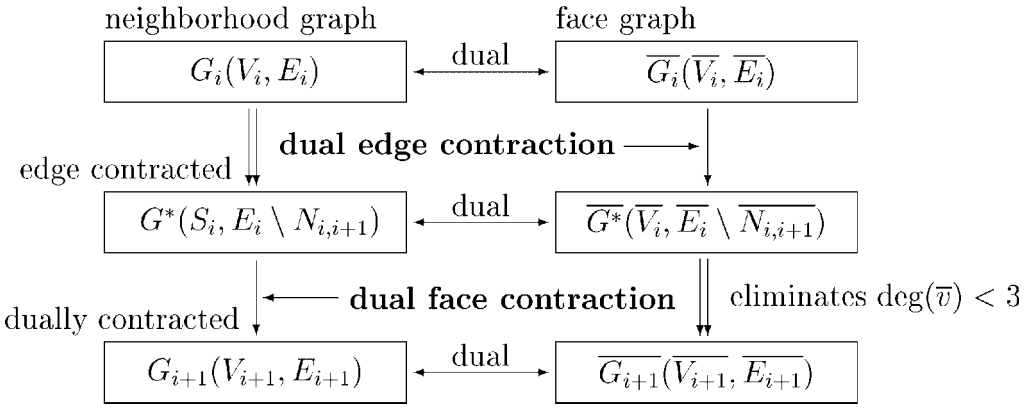

Fig. 12. Dual Graph Contraction: $(G_{i|1}, \overline{G_{i|1}}) = C[(G_i, \overline{G_i}), (S_i, N_{i,i|1})]$

the vertices across the different levels. Each vertex at the top level corresponds to a connected set of vertices in the level below. Each of those corresponds to another connected set of vertices in the level below and so on. The union of all the sets of vertices in the base level that are derived from one vertex at the top level form the receptive field of this vertex.

2. Formal Definition

Figure 12 summarizes the two basic steps: dual edge contraction and dual face contraction. The base of the pyramid consists of the pair of dual image graphs $(G_0, \overline{G_0})$. A new (reduced) level $i+1$ is computed from level i by

1. selecting the contraction kernels $(S_i, N_{i,i|1})$,
2. dually contracting the selected edges, and
3. removing redundant multi-edges and self-loops (dual face contraction).

Connectivity of surviving vertices is preserved if the contraction kernels satisfy the following definition of an irregular pyramid (see also Kropatsch 1994 [Def.5]):

Definition 1. *In a pair of dual image graphs* $(G_i(V_i, E_i), \overline{G_i}(\overline{V_i}, \overline{E_i}))$, *following* **decimation parameters** $(S_i, N_{i,i|1})$ *determine the contracted graphs* $(G_{i|1}, \overline{G_{i|1}})$: *a subset of* **surviving vertices** $S_i = V_{i|1} \subset V_i$, *and a subset of* **primary non-surviving edges**[4] $N_{i,i|1} \subset E_i$. *The decimation parameters* $(S_i, N_{i,i|1})$ *must be a subgraph of* G_i *and do not contain any circuit, e.g.* $(S_i, N_{i,i|1})$ *is a forest. The relation between the two pairs of dual graphs,* $(G_i, \overline{G_i})$ *and* $(G_{i|1}, \overline{G_{i|1}})$, *as established by dual graph contraction with decimation parameters* $(S_i, N_{i,i|1})$ *is expressed by function* $C[.,.]$:

$$(G_{i|1}, \overline{G_{i|1}}) = C[(G_i, \overline{G_i}), (S_i, N_{i,i|1})] \quad (1)$$

The connected components of the decimation parameters are called contraction kernels.

B. The Graph Pyramid and Equivalent Contraction Kernels

A contraction kernel collects all edges that can be contracted independently of each other without destroying the connectivity structure of the graph. Since

[4]Secondary non-surviving edges are removed during dual face contraction.

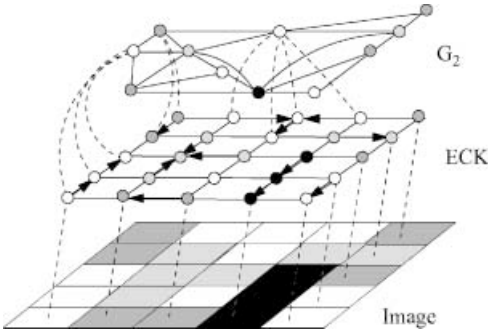

Fig. 13. Equivalent contraction kernels (ECK) contract G_0 directly into G_2

the contraction operation is forbidden for self-loops the set of edges involved in such a sequence of contractions must not contain a circuit. Thus the set of edges involved in such a contraction may be encoded by a tree, or, a collection of non-overlapping trees spanning the given input vertices: a *spanning forest*.

Repeated dual graph contraction builds a stack of successively smaller graphs: the graph pyramid. Let us denote the contraction of a graph G_0 by a contraction kernel $N_{01} \subset E_0$ by $G_1 = G_0/N_{01}$ and the subsequent contraction by $G_2 = G_1/N_{12}$. Then there exists an *equivalent contraction kernel* $N_{02} \subset E_0$ that creates the same result in a single step: $G_2 = G_0/N_{02}$. Our example pyramid in Fig. 11 was built using two contraction levels. The same top level graph can be derived in a single step using equivalent contraction kernels as shown in Fig. 13. Note that some kernels contain two edges to be contracted one after the other. Consequently more parallel steps are needed to compute the result. Conversely, a contraction kernel may be decomposed into two smaller ones. The successive application of the resulting contraction kernels produces the same result as the application of the initial one. The complete formalism is described in (Kropatsch 1997a).

Equivalent contraction kernels relate the data at the base level directly with any higher level in the graph pyramid and can be seen as the receptive field of the corresponding pyramidal cell. This property has been used in Fig. 14 to show some of the 'internal' structures of the graph pyramid on the famous picture of (Bister et al. 1990) (Fig. 14a). Some of the top levels have been down-projected to the base graph by means of their equivalent contraction kernels. Since the receptive field corresponding to a lower level vertex is always included in the receptive field of its parent vertex also the boundaries of the receptive fields form such an inclusion hierarchy. Figure 14b shows the top level contours which follow the boundary between black and white in the original image but also some internal

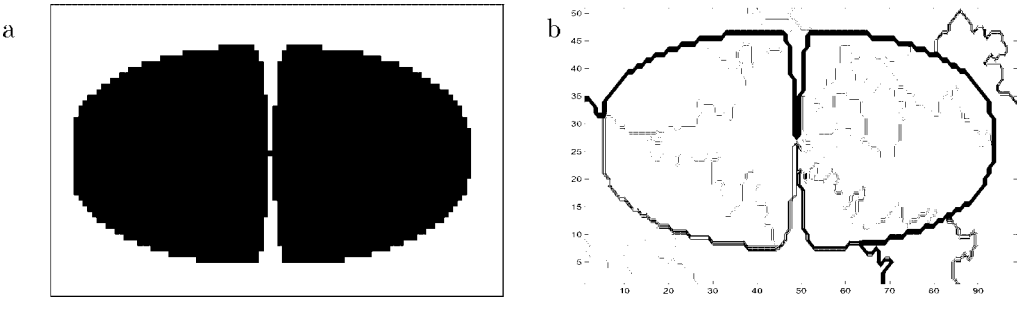

Fig. 14. Visualizing the graph pyramid. *a* A binary image, *b* boundaries of some receptive fields

kernels which have been chosen stochastically in absense of any other control.

Since also the contents of a pyramidal cell is the result of applying a series of reduction functions there is also the equivalent way to compute the same value directly from the base. Although this is not more efficient than the iteration through the pyramid levels it offers the possibility to invert this process: given a function to compute a value for a complex decision how can this computation be decomposed into a series of local reduction functions which can be efficiently computed using the pyramid? It is clear that not all functions can be split up in such a way, but there are many that offer this possibility and are subject of current research.

C. The Domain of All Graph Pyramids

Another question concerns the functionality of a given sensor arrangement if it is equipped with the flexibility of the interconnection network of the graph pyramid: Can I compute a given complex decision by adjusting the selection criteria and the reduction functions? What is the domain of all decision functions that can be computed?

These far reaching questions have two components: a structural component and a functional component. The structural component identifies all sensors that influence the function. The functional component involves the coding of the information and the definition of functions and methods transforming a structured

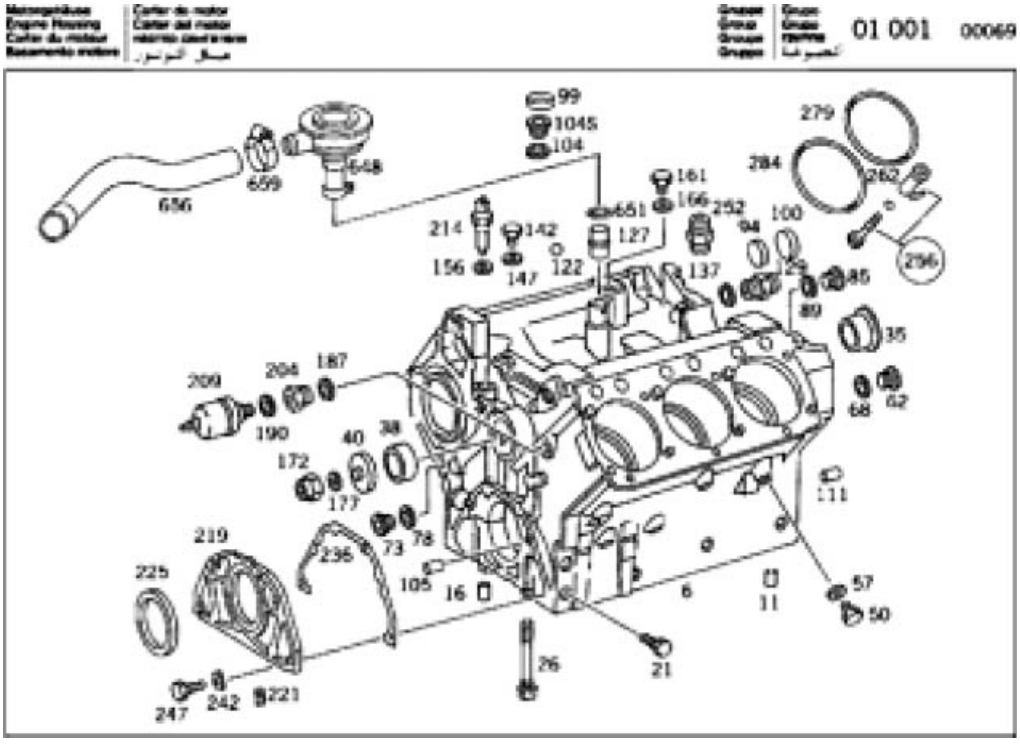

Fig. 15. Technical drawing analyzed by graph pyramid

set of values and codes into a more general value or code. The structural component can be studied through the concept of equivalent contraction kernel, the functional component may go beyond the scope of linear filters as in the equivalent weighting functions of (Burt and Adelson 1983) and may involve symbolic representations and learning concepts.

V. Conclusion

We summarize the conceptual framework which has been set up to perform dual graph contraction. Contraction is controlled by kernels that can be combined in many ways. Data and relations considered important for interpreting the visual input are selected to survive a repeated data reduction which, at the same time, preserves topological properties among the surviving parts. The hierarchy created in this way adapts its structure to the sensor data while integrative measures can be efficiently computed within the (sub-) hierarchy of the receptive field.

Preliminary experiments with graph pyramids have been successful in several areas, e.g., connected component labeling (Macho and Kropatsch 1995); segmentation (Kropatsch and BenYacoub 1996); '2x on a curve' (Kropatsch 1997b); line images like that shown in Fig. 15 (Burge and Kropatsch 1999); matching (Pailloncy et al. 1998); isolating moving objects from background (Kropatsch 1999); generalization preserving monotonic landscape properties (Glantz et al. 1999).

Graph pyramids have been introduced as an efficient model for visual information processing. Efficiency in graph pyramids is achieved through vertical pathways connecting the apex with the sensors in the base through the pyramidal hierarchy across the

$$\log\left(\textit{diameter}(\text{base graph})\right)$$

number of (horizontal) levels (Kropatsch 1997b).

References

Bister M, Cornelis J, Rosenfeld A (1990) A critical view of pyramid segmentation algorithms. Pattern Recogn Lett, 11(No. 9), pp 605–617

Burge M, Kropatsch WG (1999) A minimal line property preserving representation of line images. Computing, Devoted Issue on Image Processing, 62, pp 355–368

Burt PJ, Adelson EH (1983) The laplacian pyramid as a compact image code. IEEE Transactions on Communications Vol. COM-31(No.4), pp 532–540

Gareth AJ, Singerman D (1978) Theory of maps on orientable surfaces. Vol. 3. London Mathematical Society

Glantz R, Englert R, Kropatsch WG (1999) Representation of image structure by a pair of dual graphs. In: Kropatsch WG, Jolion J-M (eds), 2nd IAPR-TC-15 workshop on graph-based representation. Österreichische Computer Gesellschaft, OCG-Schriftenreihe Band 126, pp 155–163

Jolion J-M, Kropatsch WG (eds) (1998) Graph based representations in pattern recognition. Computing, Suppl 12. Springer-Verlag Wien New York

Jolion J-M, Rosenfeld A (1994) A Pyramid Framework for early Vision. Kluwer Academic Publishers

Jolion J-M, Kropatsch WG, Vento Mario (eds) (2001) Graph-based representations in pattern recognition, GbR 2001. CUEN. ISBN 88 7146 579–582

Kropatsch WG (1994) Building irregular pyramids by dual graph contraction. Tech rep PRIP-TR-35. Institute f. Automation 183/2, Pattern Recognition and Image Processing Group, TU Wien, Austria. Also available through http://www.prip.tuwien.ac.at/ftp/pub/publications/trs/tr35.ps.gz

Kropatsch WG (1997a) Equivalent contraction kernels to build dual irregular pyramids. Advances in Computer Vision, pp 99–107, Springer series Advances in Computer Science

Kropatsch WG (1997b) Property preserving hierarchical graph transformations. In: Arcelli C, Cordella LP, Sanniti di Baja G (eds) Advances in Visual Form Analysis. World Scientific Publishing Company, pp 340–349

Kropatsch WG (1999) How useful is structure in motion? In: Chetverikov D, Szirányi T (eds) Fundamental Structural Properties in Image and Pattern Analysis 1999. Österreichische Computer Gesellschaft, OCG-Schriftenreihe Band 130, pp 35–45

Kropatsch WG, BenYacoub S (1996) Universal segmentation with p*IRR*amids. In: Pinz A (ed) Pattern Recognition 1996, Proc of 20th ÖAGM workshop R. Oldenburg, OCG-Schriftenreihe, Band 90, pp 171–182

Kropatsch WG, Brun L (2000) Hierarchies of combinatorial maps. In: Svoboda T (ed) CPRW2000, Proceedings of the Czech Pattern Recognition Workshop. Peršlák, CZ: Czech Pattern Recognition Society Praha. ISBN 80-238-5215-9, pp 131–137

Kropatsch WG, Jolion J-M (eds) (1999) 2nd IAPR-TC-15 workshop on graph-based representation. Österreichische Computer Gesellschaft Band 126

Lienhardt P (1989) Subdivisions of *n*-dimensional spaces and *n*-dimensional generalized maps. In: Mehlhorn K (ed) Proceedings of the 5th annual symposium on computational geometry (SCG '89). Saarbrücken, FRG: ACM Press, pp 228–236

Macho H, Kropatsch WG (1995) Finding connected components with dual irregular pyramids. In: Solina F, Kropatsch WG (eds) Visual Modules, Proc of 19th ÖAGM and 1st SDVR workshop R. Oldenburg, OCG-Schriftenreihe Band 81, pp 313–321

Marr D (1982) Vision. San Francisco: Freeman WH

Pailloncy J-G, Kropatsch WG, Jolion J-M (1998) Object matching on irregular pyramid. In: Jain AK, Venkatesh S, Lovell BC (eds) 14th International Conference on Pattern Recognition, Vol. II. IEEE Comp Soc, pp 1721–1723

Pizlo Z, Salach-Golyska M, Rosenfeld A (1997) Curve detection in a noisy image. Vision Res 37(9): 1217–1241

Rosenfeld A, Nakamura A (1997) Local deformations of digital curves. Pattern Recogn Lett 18: 613–620

Thulasiraman K, Swamy MNS (1992) Graphs: Theory and Algorithms. New York, USA: Wiley-Interscience

Uhr L, Schmitt L (1984) The several steps from icon to symbol using structured cone/pyramids. In: Rosenfeld A (ed) Multiresolution Image Processing and Analysis. Berlin, Heidelberg, New York, Tokyo: Springer Verlag, pp 86–100