# Chapter 1

# Hierarchical Graph Encodings

LUC BRUN

GREYC
ENSICAEN
Caen, France
Email: luc.brun@greyc.ensicaen.fr

WALTER KROPATSCH

Pattern Recognition and Image Processing Group
Technical University of Vienna
Vienna, Austria
Email: krw@prip.tuwien.ac.at

Here is the Abstract we provided, just for recall, we should remove it at the end: The first part of this chapter briefly describes the main properties and drawbacks of regular pyramids. Based on the limitations of such pyramid the notion of irregular pyramid is introduced in the second part. We focus our attention on three types of pyramids: simple graphs (without loops nor double edges), dual graphs and combinatorial pyramids. These three models are presented using a generic notion of kernel. The main concepts are first introduced using simple graph pyramids. After a presentation of this model together with its main limitations introduce the main properties of dual graph pyramids. A presentation of the combinatorial pyramid model conclude the second part of this chapter. A brief introduction to the extension of such models to n dimensional spaces is also provided. The third and last part of this chapter presents the main construction schemes of

such hierarchical models. More precisely, we introduce, the notions of Maximal independent set, maximal independent edge set and data driven decimation.

# Contents

## 1.1 Introduction

Regular image pyramids have been introduced in 1981/82 [1] as a stack of images with decreasing resolutions. Such pyramids present interesting properties for image processing and analysis such as [2]: The reduction of noise, the processing of local and global features within the same frame, the use of local processes to detect global features at low resolution and the efficiency of many computations on this structure. Regular pyramids are usually made of a very limited set of levels (typically $\log(n)$ where $n$ is the diameter of the input image). Therefore, if each level is constructed in parallel from the level below, the whole pyramid may be built in $O(\log(n))$ steps. However, the regular structure of such pyramids induce several drawbacks such as the shift dependence problem and the limited number of regions which may be encoded at a given level.

Irregular pyramids overcome these negative properties while keeping the main advantages of their regular ancestors. Such pyramids are defined as a stack of successively reduced graphs. Each vertex of such a pyramid being defined from a connected set of vertices in the graph bellow. Typically, such graphs encode the pixel's neighborhood, the region adjacency or a particular semantic of image objects. Since 2D images correspond to a sampling of the plane, such graphs are embeded in the plane by definition.

The construction of such pyramids rises two questions: 1) Which properties of the plane are captured by a specific graph model ? 2) Given a graph model, how to build efficiently each level of the pyramid from the one bellow ? This chapter provides answers to both questions together with an overview of hierarchical models. We first present in Section 1.2 the regular pyramid framework. The different types of irregular pyramids are then presented in Section 1.4 together with their main properties. We conclude this chapter, in Section 1.3, by a panorama of the different methods used to build irregular pyramids in parallel.

## 1.2 Regular Pyramids

A regular pyramid is defined as a sequence of images with and exponentially decreasing resolution. Each image of such a sequence is called a *level* of the pyramid. The first level, called the *base* of the pyramid corresponds to the original image while the *top* of the pyramid corresponds usually to a single pixel whose value is a weighted mean of pixels belonging to the base. Using image neighborhood relationships, a *reduction window* connects any pixel of the pyramid to a set of pixels defined at the level bellow. Any pixel of a pyramid is considered as the father of the pixels belonging to its reduction window (filled circles in Fig. 1.1). Conversely, all pixels within a reduction window are considered as the *children* of the pixel associated to this reduction window in the above level. The father/child or hierarchical relationship may be extended between any two level using the transitive closure of the relationship induced by the reduction window. The set of pixels in the base level image associated to a given pixel of the pyramid is called its *receptive field*. Let us, for example consider the top-left pixel in Fig. 1.1(c). Its reduction window and receptive field are respectively defined as the $2 \times 2$ and $4 \times 4$ squares located on the top left part of the image.

One additional imporant parameter of a regular pyramid is the *reduction factor* which encodes the ratio between the size of two successive images. This ratio remains the same between any two successive levels of the pyramid. A *reduction function* determines the value of any pixel of the pyramid from the value of its

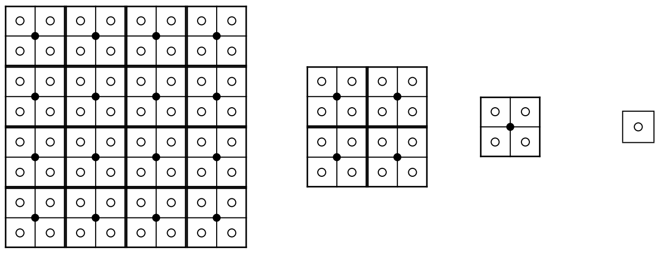


Figure 1.1: A $2\times 2/4$ regular pyramid

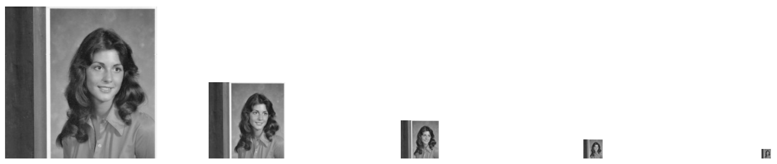


Figure 1.2: A $2\times 2/4$ pyramid whose reduction function is defined as a Gaussian centered on the center of the reduction window. The size of each image is indicated bellow.

child defining its reduction window (Fig.1.2). A regular pyramid may thus be formally defined as the ratio $N\times N/q$, where $N\times N$ represent the size of the reduction window while $q$ represent the decimation ratio. Different types of pyramid may be distinguished according to the value of $N\times N/q$:

- If $N\times N/q<1$, the pyramid is named a *non-overlapping holed pyramid*. Within such pyramid, some pixels have no parent [3] (e.g., the center pixel in Fig. 1.3(a)).
- If $N\times N/q=1$, the pyramid is called a *non overlapping pyramid without hole*(see e.g. in Fig. 1.3(b)). Within such pyramids, each pixel in the reduction window has exactly one parent [3].
- If $N\times N/q>1$, the pyramid is named an *overlapping pyramid* (see e.g. in Fig. 1.3(c)). Each pixel of such a pyramid has on average $N^2/qparents$ [1]. If each child selects one parent, the set of children in the reduction window of each parent is re-arranged. Consequently the receptive field may take any form inside the original receptive field.

Regular Pyramids have several interesting properties enumerated by Bister [2]:

1. Reduction functions are usually defined as low pass filter hence inducing a robustness against noise in high levels of the pyramid,
2. The use of the pyramid allows to readily design algorithms insensitive to the image's resolution,

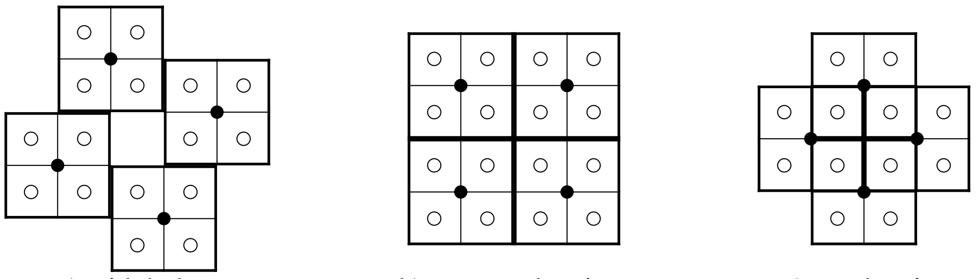


Figure 1.3: Three different types of regular pyramids.

3. Gobal properties in the base level image become local one at higher levels in the hierarchy and may thus be detected using local filters,

4. Top-down analysis of the content of a pyramid may be achieved efficiently using a divide and conquer strategy.

5. Objects to be detected may be retreived using low resolution images with a simplified image content.

Despite, these interesting properties, regular pyramids have several limitations. Indeed, the bounded size and fixed shape of the reduction window induce rigid constraints on the data structure which may not allows to readily handle the variability of image content. Regular pyramids are for example quite sensitive to small image shifts due to the fixed locations of reduction window. The fixed size of reduction windows bounds artificially the number of regions that may be encoded at a given level and does not allow to readily encode elongated objetcs.

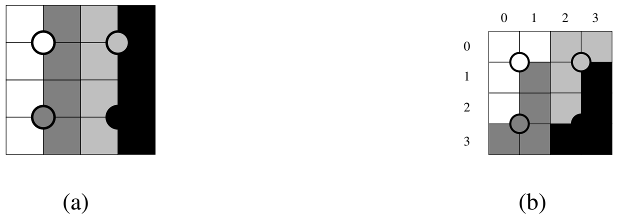


Figure 1.4: A $4 \times 4$ base level image(a) and its segmentation according to the values of pixels defined at level 1(b)

This last drawback is illustrated Fig. 1.4 using an example proposed by [2]. The base level image of this pyramid is composed of 4 columns of different color. The values of the four pixels defined at level 1 of the pyramid are represented by filled cicles. Note that the colors of these circles is choosen so as to encode all the 4 regions defined at the base level. Using a $4 \times 4/4$ image pyramid, the bottom left pixel with coordinated $(3,0)$ (Fig. 1.4(b)) can not belong to the reduction window of the white circle with coordinates $(\frac{1}{2},\frac{1}{2})$. Therefore, pixel $(3,0)$ can not be attached to its white parent and a labeling of the base level image according to labels defined at level one attach this pixel to the dark-grey label which corresponds to the labels of pixel $(2.5,0.5)$ which is the only pixel defined at level one whose reduction window contain $(3,0)$. In this case, pixel $(3,0)$ is artificially removed from the white region (Fig. 1.4(a)) due to the limited size of reduction windows. Note that the same problem occurs with pixels $(0,1),(0,3)$ and $(3,2)$ (Fig. 1.4(b)).

This phenomenon is also illustrated in Fig. 1.4 on a real image using bister [2] algorithm and a $5 \times 5/4$ pyramid. As shown in Fig. 1.4(b), several elongated regions located on the left part of the image together with the hair of girl are artificially split into several small regions.

(a) original Image

(b) segmented Image

Figure 1.5: Image segmentation of Girl test image using Bister's algorithm [2] with an overlapping $5 \times 5/4$ pyramid.

Let us finally temperate the above remarks by noting that Bister [2] and his team allways managed to adapt their regular pyramid to the class of images they have to process. However such an adaptation choice of the reduction window, reduction factor, reduction functionn...should be performed manually for each image class. We can additionally note that some drawbacks of regular pyramids may become advantages on some specific applications. For example the fact that regular pyramid do not readily allow to encode elongated objects may become an advantage in fields where objects to be detected are known to be compact. Such an example of application is provided by Rosenfeld [4] which use a regular pyramid in order to detect tanks in infrared images. Within this particular context tanks are known to be compact and may detected as single pixels at some levels of the pyramid.

## 1.3 Irregular Pyramids Parallel reduction schemes

Irregular pyramids have been first proposed by Meer and Montanvert [5] in order to overcome negative properties of their regular ancestors while keeping their main advantages. More particularly, two important

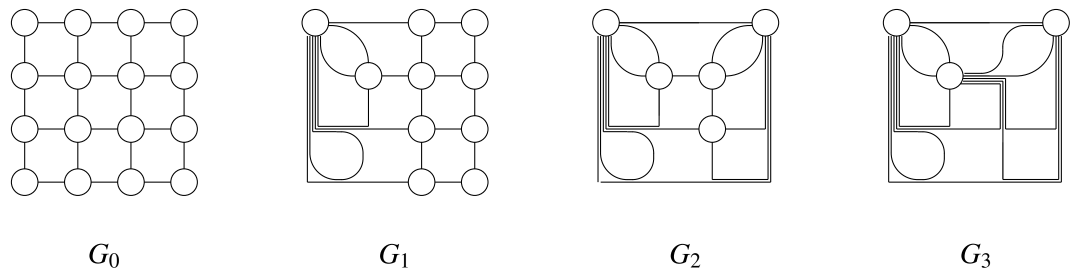


Figure 1.6: First levels of an irregular pyramid using the 4-adjacency of pixels.

advantages of regular pyramids over non-hierarchical image processing algorithms should be preserved:

1. The bottom-up computation of any pyramid may be performed in parallel, each pixel computing its value from its child independently.

2. Due to the fixed size of the decimation ratio, the height of a pyramid is equal to a log of the image size. Therefore, using a parallel computation scheme any pyramid may be computed in $O(\log(|I|))$, where $|I|$ denotes the size of the base level image.

On the other hand, one of the main drawbacks of Regular Pyramids is the fixed shape of reduction windows which do not allow to readily adapt the pyramid to the data and the fixed neighborhood of each pixel which do not allow to preserve adjacency relationships along the pyramid. In order to overcome these drawbacks, Meer and Montanvert [5], defined irregular pyramids as stacks of successively reduced graphs ($G_0, G_1, G_2, G_3$ in Fig. 1.6). Such pyramids may adapt the shape of the reduction window to the data and preserve the adjacency relationships between regions thought graph encoding since a vertex may have an arbitrary number of neighbors. In reference to their regular ancestors, the decimation ratio of irregular pyramids is usually defined as the mean value of the ratio between the number of vertices of two successive graphs $\left(\frac{|\mathcal{V}_l|}{|\mathcal{V}_{l+1}|}\right)$ when this last term is approximately constant along the pyramid. The construction of a graph $G_{l+1} = (\mathcal{V}_{l+1}, \mathcal{E}_{l+1})$ from $G_l = (\mathcal{V}_l, \mathcal{E}_l)$, is performed thought the following steps:

1. Define a partition of $\mathcal{V}_l$ into a set of connected components. A vertex of $G_{l+1}$ is created and attached to each connected component of this partition. The connected component attached to each vertex of $G_{l+1}$ is called its *reduction window* in reference to regular pyramids.

2. Define adjacency relationships between vertices of $G_{l+1}$ from adjacency relationships between the connected components of $G_l$'s partition.

This section is devoted to parallel reduction schemes which are uniquely concerned by the first step of the above construction scheme, the second step being detailed in Section 1.4. Note that the number of connected components defining a partition of $G_l$, defines the cardinal of $\mathcal{V}_{l+1}$ and hence the decimation ratio of the irregular pyramid. Several parallel methods have been proposed to insure large values of the decimation ratio $\left(\frac{|\mathcal{V}_l|}{|\mathcal{V}_{l+1}|}\right)$, all these methods being based on the notion of *Maximal independent set*.

### 1.3.1 Maximal independent set

Let us consider a finite or enumerable set $\mathcal{X}$ provided with a symmetric neighborhood function $\mathcal{N} : \mathcal{X} \rightarrow \mathcal{P}(X)$, where $\mathcal{P}(\mathcal{X})$ denotes the powerset of $\mathcal{X}$. An independent set of $\mathcal{X}$ is a subset $\mathcal{I}$ of $\mathcal{X}$ such that no element of $\mathcal{I}$ are related by the neighborhood function $\mathcal{N}(.)$:

$$\forall (x,y) \in \mathcal{I}^2 \; x \not\in \mathcal{N}(y) \tag{1.1}$$

An independent set is said to be maximal if it is not the subset of any independent set. For example, the set of even number, is a maximal independent set over the set of natual integers provided with the neighborhood relationship : $\mathcal{N}(n) = \{n-1, n+1\}$. Note that, both even and odd integers, define a maximal independent set using this neighborhood relationship. A maximal independent set is thus not unique. Moreover, given a maximal independent set $\mathcal{I}$, any element $x$ in $\mathcal{X} - \mathcal{I}$ must be adjacent to some element of $\mathcal{I}$, otherwize $x$ could be added to $\mathcal{I}$ hence violating the maximal property. Conversely, if $\mathcal{I}$ is a non maximal independent set, at least one element $x$ may be added to $\mathcal{I}$. This element is not adjacent to any element of $\mathcal{I}$ due to condition 1.1. The maximality of an independent set may thus be characterized by:

$$\forall x \in \mathcal{X} - \mathcal{I} \; \exists y \in \mathcal{I} \,|\, x \in \mathcal{N}(y) \tag{1.2}$$

Within the irregular pyramid framework elements of a maximal independent set are usually called *surviving elements*. Condition 1.1, states thus that two adjacent elements can not both survives. On the other hand, condition 1.2, states that any non surviving element should be adjacent to a surviving one. A maximal independent set may be interpreted as a sub sampling of the initial set $\mathcal{X}$ according to the neighborhood relationship $\mathcal{N}$. Let us for example, consider the case where $\mathcal{X}$ corresponds to the set of pixels of an image $\mathcal{I}$, while the neighborhood relationship encodes the 8 connectivity of pixels. A maximal independent set, may then be achieved by selecting one pixel every two lines and two columns. One may easily check in Fig. 1.7 that adjacent surviving pixels (■) are not adjacent (condition 1.1) and that any surviving pixel (□) is adjacent to at least one surviving pixel(condition 1.2).

The remaining of this chapter will be devoted to finite sets. Given such a finite set $\mathcal{X}$, an independent set will be called *maximum* if its cardinal is maximum over all possible maximal independent sets. Moreover, in such a case, neighborhood relationships may be encoded by a graph $G = (\mathcal{X}, \mathcal{E})$, where $(u,v) \in \mathcal{X}^2$ belongs to $\mathcal{E}$ if and only if $u \in \mathcal{N}(v)$.

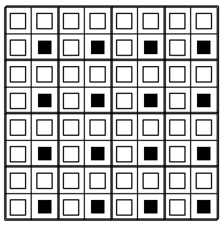


Figure 1.7: A maximal independent set (■) over a planar sampling grid with the 8 neighborhood.

### 1.3.2 Maximal independent vertex set

A maximal independent vertex set (MIVS) of a graph $G_l = (\mathcal{V}_l, \mathcal{E}_l)$ is defined as a maximal independent set over the set $\mathcal{V}_l$ using the neighborhood relationship induced by $\mathcal{E}_l$. If we denote the maximal independent set by $\mathcal{V}_{l+1}$, conditions 1.1 and 1.2 may be written as follows:

$$\forall v \in \mathcal{V}_l - \mathcal{V}_{l+1}\ \exists v' \in \mathcal{V}_{l+1} :\ (v, v') \in \mathcal{E}_l \tag{1.3}$$

$$\forall (v, v') \in \mathcal{V}_{l+1}^2 :\ (v, v') \notin \mathcal{E}_l \tag{1.4}$$

The construction scheme of such a maximal independent set was introduced by Meer [6] as an iterative stochastic process based on the outcome of a random variable attached to each vertex and uniformly distributed over $[0,1]$. Any vertex corresponding to a local maximum of the random variable is then considered as a surviving vertex. One iteration of this process allows to satisfy condition 1.3 and hence to obtain an independent set, since to adjacent vertices can not both correspond to local maxima. However, one vertex not corresponding to a local maximum may have no surviving vertices in its neighborhood. In other terms, such a process may lead to a non maximal independent set. Such a configuration is illustrated on Fig. 1.8 where values inside each vertex represent the outcome of the random variable. Vertices with value 9 represent local maxima of the random variable and are thus selected as survivors. Vertices 7 and 8 adjacent to a surviving vertex can not be selected as survivors due to condition 1.3. Vertex 6 does not correspond to a maxima of the random variable. This vertex should nevertheless be selected as a survivor in order to fulfill condition 1.4 and obtain a maximal independent set.

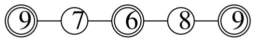


Figure 1.8: Definition of a maximal independent vertex set on a $1D$ graph. Surviving vertices are represented by double circles.

We should thus iterate the selection of local maxima until both conditions 1.3 and 1.4 are satisfied. Such an iterative process requires to attach three variables $x_i, p_i, q_i$ to each vertex $v_i \in \mathcal{V}_l$. Variable $x_i$ encodes the outcome of the random variable attached to $v_i$ while $p_i$ and $q_i$ correspond to boolean variables whose values encode the following states:

- If $p_i$ is true, $v_i$ is considered as a surviving vertex,
- If $q_i$ is true, $v_i$ is a *candidate* which may become a surviving vertex at some further iteration. Conversely, if $q_i$ is false, $v_i$ is considered as a non surviving vertex.

Let $(p_i^{(k)})_{k\in\{1,\dots,n\}}$ and $(q_i^{(k)})_{k\in\{1,\dots,n\}}$ denote the values taken by variables $p_i$ and $q_i$ along the iteration of our iterative algorithm. Variable $p_i$ and $q_i$ are initialized as follow:

$$\begin{array}{rcl} p_i^{(1)} & = & x_i = max_{j\in\mathcal{N}(v_i)}\{x_j\} \\ q_i^{(1)} & = & \bigwedge_{j\in\mathcal{N}(v_i)} \overline{p_j}^{(1)} \end{array} \tag{1.5}$$

where $\mathcal{N}(v_i)$ denotes the neighborhood of $v_i$ and $\bigwedge$ corresponds to the logical "and" operator. We suppose by convention that $v_i \in \mathcal{N}(v_i)$.

Equation 1.5 states that a vertex survives $(p_i = true)$ if it corresponds to a maxima of the random variable. One vertex is a candidate $(q_i = true)$ if it is not already a survivor and if none of its neighbors survive.

The iterative update of the predicate $p_i$ and $q_i$ is performed using the following rules:

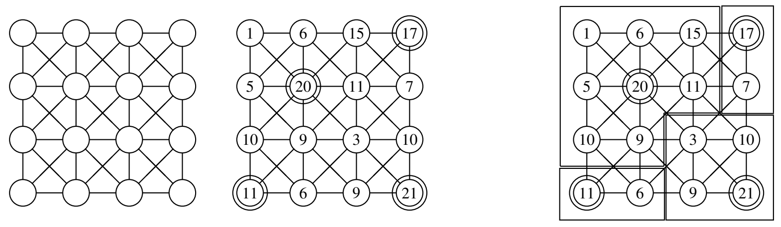
$$\begin{array}{rcl} p_i^{(k+1)} & = & p_i^{(k)} \vee \left(q_i^{(k)} \wedge\ x_i = max_{j\in V(v_i)}\{q_j^{(k)} x_j\}\right) \\ q_i^{(k+1)} & = & \bigwedge_{j\in V(v_i)} \overline{p_j}^{(k+1)} \end{array} \tag{1.6}$$

where $\vee$ denotes the logival "or" operator.

One vertex surviving at iteration $k$ will thus remain a survivor at further iterations. Moreover, a candidate may become a survivor if it corresponds to a local maxima of all the candidate in its neighborhood at some iteration. Finally, a candidate remains in this state ($q_i = true$) if it does not become a survivor and if none of its neighbor is selected as a survivor.

Note that the set of vertices such that $p_i = true$ defines an independent set at any step of this iterative algorithm. Each step decreases the number of candidates and increases the number of survivors until no more survivors may be added. The set of survivors then define a maximal independent set. Let us additionally note that this process is purely local. Indeed, for each iteration, each vertex updates the values of $p_i$ and $q_i$ according to the one of its neighbors. Such a process may thus be easily encoded on a parallel machine. Fig. 1.9 (b) represents the outcomes of the random variable on a graph defined from the $4\times 4$ 8-connected sampling grid (Fig. 1.9 (a)). Surviving vertices are surrounded by an extra circle.

Jolion [7] improves the adaptability of the above decimation process by defining surviving vertices as local maxima of an interest operator. For example, within the segmentation framework, Jolion defines the



(a) $G$ (b) MIVS (c) Reduction windows

Figure 1.9: Construction of a Maximal Independent Vertex Set using a random variable on a graph $G$ encoding a $4 \times 4$ 8 connected sampling grid(a). The outcome of the random variable are displayed inside each vertex. Surviving vertices are surrounded by nested circles(b). Reduction windows associated to surviving vertices (c).

operator of interest as a decreasing function of the gray level variance computed in the neighborhood of each vertex. This operator provides a location of surviving vertices in homogeneous regions.

#### Definition of reduction window

Given the set of surviving vertices, constraint (1.3) of a MIVS insures that each non-surviving vertex is adjacent to at least one surviving vertex. The selection of a surviving parent by each non surviving vertex may be performed using various heuristics. Meer [6] and Montanvert [5] attach a random variable to each vertex, and connect each non surviving vertex to its surviving neighbor with the greatest value of the random variable. Jolion [7] uses a contrast measure such as the gray level difference, to link each non-surviving vertex to its least contrasted surviving neighbor. Each surviving vertex is thus the father of all non-surviving vertices attached to it. Fig. 1.9(c) represents the reduction window defined from Fig. 1.9(b) by connecting each non surviving vertex to its surviving neighbor of greatest value.

Given the partition of $\mathcal{V}_l$ into a set of reduction window with their associated suvivors $\mathcal{V}_{l+1}$, the reduced graph $G_{l+1} = (\mathcal{V}_{l+1}, \mathcal{E}_{l+1})$ is defined using various heuristics defined in Section 1.4.

### 1.3.3 Data driven decimation Process

As mentioned in Section 1.3.2, a vertex selected as a survivor remains in this state until the end of the iterative construction scheme of a MIVS (equation 1.6). Moreover, the neighbors of such a survivor are classified as non candidates and remain also in this state until the end of the iterative process. Therefore, one surviving vertex together with its non surviving neighbors could be reduced into a single vertex of the reduced graph as soon as it is classified as a survivor at some iteration step. However, using the construction scheme defined in Section 1.3.2, the definition of a reduced graph is based on a MIVS and requires thus that all the vertices of the graph are marked as survivors or non candidate. For important graphs this latency which is proportional to the number of iterations may be important.

The basic idea of the Data Driven Decimation Process (D3P) proposed by Jolion [8] consists to perform only one iteration of the iterative decimation process (equation 1.6) at each level of the pyramid. The two variables $p_i$ and $q_i$ defined in Section 1.3.2 keep the same meaning but are now global to the whole pyramid. These variables are initialized to true at the base level of the pyramid. We have thus :

$$p_i^1 = q_i^1 = true, \forall v_i \in \mathcal{V}_0$$

where $G_0 = (\mathcal{V}_0, \mathcal{E}_0)$ encodes the base of the pyramid.

The update of variables $p_i$ and $q_i$ between each level is performed using the following equations:

$$\begin{array}{rcl} p_i^{(k+1)} & = & \left((p_i^{(k)} \vee q_i^{(k)}) \wedge x_i = max_{j \in \mathcal{N}_k(v_i)} \{q_j^{(k)} x_j\}\right) \\ q_i^{(k+1)} & = & \bigwedge_{j \in \mathcal{N}_k(v_i)} \overline{p_j}^{(k+1)} \wedge (\mathcal{N}_k(v_i) \neq \{v_i\}). \end{array} \quad (1.7)$$

In other terms, a vertex $v_i$ is selected as a survivor if it was a survivor or a candidate in graph $G_k$ and if it corresponds to a maximum of the random variable among the candidates in $\mathcal{N}_k(v_i)$. A vertex of $G_k$ becomes a candidate if it is not adjacent to any surviving vertex and if it is not isolated ($\mathcal{N}_k(v_i) \neq \{v_i\}$).

The set of surviving vertices determined by equation 1.7 defines an independent set over $G_k$. This set is however, usually non maximal. More precisely, we may decompose the set $\mathcal{V}_k$ into three subsets based on variables $p_i^{(k+1)}$ and $q_i^{(k+1)}$: Survivors ($p_i^{(k+1)} = true$), candidates ($q_i^{(k+1)} = true$) and non candidates ($q_i^{(k+1)} = false$). Non candidate vertices are adjacent to at least one survivor (equation 1.7). These vertices together with surviving vertices are grouped into reduction windows reduced into a single vertex in graph $G_{k+1}$. Candidate vertices represent the set of vertices which would have been classified as survivors or non survivors at further iteration steps using the iterative construction scheme of a MIVS (equation 1.6). Using D3P construction scheme, the classification of such vertices is delayed to higher levels of the pyramid. The set of vertices defined at level $k+1$ is thus equal to the set of survivors together with the set of candidates:

$$\mathcal{V}_{k+1} = \{v_i \in \mathcal{V}_k \mid p_i^{(k+1)} \vee q_i^{(k+1)}\}.$$

The set $\mathcal{E}_{k+1}$ is defined according to various heuristics defined in Sections 1.4.3 and 1.4.4. Equation 1.7 is then applied on the reduced graph $G_{k+1} = (\mathcal{V}_{k+1}, \mathcal{E}_{k+1})$ until the top of the pyramid.

Let us recall (Section 1.3.2) that Jolion [7] defines surviving vertices as local maxima of an interest operator. One of main advantage of the D3P is that vertices corresponding to strong local maxima of the interest operator are detected and merged with their non surviving neighbors at low levels of the pyramid. Remaining candidates which correspond to more complex configurations are detected at higher levels of the pyramid where the content of the graph is already simplified hence simplifying the decision process. For example, within a segmentation scheme, the interest operator may detect the center of homogeneous regions which grow at low levels of the pyramid in order to facilitate the agretation of vertices encoding pixels on the boundary of two regions. Let us additionally note, that this asynchronous decimation process is consistent with some psychovisual experiments [9] showing our brain use asynchronous processes.

### 1.3.4 Maximal independent edge set

Within the MIVS framework (Section 1.3.2) two surviving vertices can not be adjacent. The number of surviving vertices defining a MIVS is thus highly dependent of the connectedness of the graph. Moreover, using the stochastic construction scheme of a MIVS defined by Meer [6] and Montanvert [5] one vertex is defined as a survivor if it corresponds to a local maxima of a random variable. The probability that a vertex survives is thus relative to the size of its neighborhood. Adjacency relationships have thus an influence on:

1. the heigth of the pyramid,
2. the number of iterations required to build each level of the pyramid from the previous one.

Several experiments performed by Haximusa et al. [**?**] have shown that the mean degree of vertices increases along the pyramid. This increasing connectivity induces the selection of a decreasing number of surviving and hence decreases the decimation ratio along the pyramid.

Such a decreases of the decimation ratio increases the height of the pyramid and violate thus one of the two main properties of Regular pyramids that Irregular pyramids should preserve (Section 1.3): The logarithmic height of the pyramid according to the size of its base.

In order to correct this important drawback Haximusa et al. [**?**] propose to base the decimation of a pyramid on the construction of a forest $K$ of $G$ such that:

Cond. I : Any vertex of $G$ belongs to exactly one tree of $K$,

Cond. II : Each tree is composed of at least two vertices.

Each tree of $K$ defines a reduction window on which a surviving vertex is selected. Remaining vertices of the tree are considered as non surviving vertices. Condition Cond. I insures that each vertex is classified either as a survivor or as a non survivor. Condition Cond. II insures that the number of vertices decreases by a factor at least 2 hence providing a reduction factor at least equal to 2.

The first step toward the construction of the forest $K$ is based on the notion of *Maximal Independent Edge Set*(MIES). A maximal independent edge set is a maximal independent set defined on the set of edges $\mathcal{E}$ of a graph $G = (\mathcal{V}, \mathcal{E})$ using the following neighborhood relationship on $\mathcal{E}$:

$$\forall (e_{uv}, e_{xy}) \in \mathcal{E}^2 \; e_{uv} \in \mathcal{N}(e_{xy}) \text{ iff } \{u,v\} \cap \{x,y\} \neq \emptyset \tag{1.8}$$

where $e_{u,v}$ denote an edge between vertices $u$ and $v$. Using the above equation, two edges are said to be neighbors if they are incident to a same vertex.

A maximal independent edge set corresponds to a maximal independent vertex set on the *edge graph* $G = (\mathcal{E}, \mathcal{E}')$ where $\mathcal{E}'$ encodes the neighborhood relationship defined by equation 1.8 [10]. The set of surviving edges defined by this decimation process corresponds to a *maximal matching* of the initial graph $G = (\mathcal{V}, \mathcal{E})$ (Definition 1 and Figure 1.10).

**Definition 1**

*Given a graph $G = (\mathcal{V}, \mathcal{E})$, a set $\mathcal{C} \subset \mathcal{E}$ is a matching if none of the edges of $\mathcal{C}$ are incident to a same vertex.*

- *A matching is said to be maximal if no edge may be added to set without violating the matching property.*
- *A maximum matching is a matching that contains the largest possible number of edges.*
- *A matching $\mathcal{C}$ is perfect is any vertex is incident to an edge of $\mathcal{C}$. Any perfect matching is maximum.*

*A vertex incident to an edge of a matching is said to be saturated. Otherwize, it is unsaturated.*

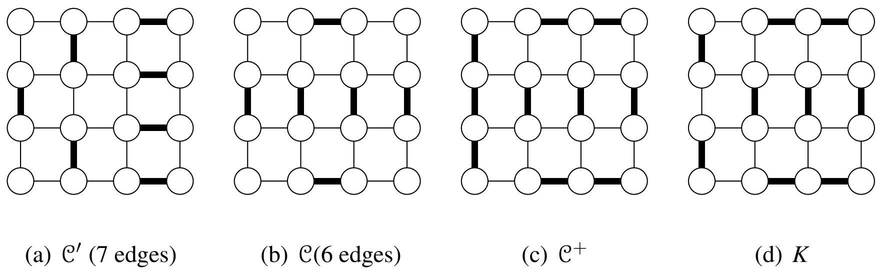

(a) $\mathcal{C}'$ (7 edges) (b) $\mathcal{C}$(6 edges) (c) $\mathcal{C}^+$ (d) $K$

Figure 1.10: Two maximal matching with different number of edges (a,b). Surviving edges are represented by thick lines. The augmented set $\mathcal{C}^+$(c) and the final forest $K$(d)

Note that a maximal matching $\mathcal{C}$ does not contain any loop and corresponds thus to a forest of the graph $G$. However, $\mathcal{C}$ is usuall not perfect (Definition 1) and violates thus Condition Cond. I since unsaturated vertices do not belong to any tree of our forest. Les us consider such an unsaturated vertex $v$ and a non self loop edge $e_{v,w}$ incident to $v$. The matching $\mathcal{C}$ being maximal, $w$ must be incident to one edge of $\mathcal{C}$. The set $\mathcal{C} \cup \{e_{v,w}\}$ is still a forest of $G$ and contains the previously unsatured vertex $v$. Let us consider the $\mathcal{C}^+$, obtained from $\mathcal{C}$ by adding edges connecting unsaturared vertices to $\mathcal{C}$. The set $\mathcal{C}^+$ is no mor a matching. However, by construction, $\mathcal{C}^+$ is a spanning of forest of $G$ composed of tree of depth 1 or 2. Trees of depth 2 may be decomposed in two trees of depth 1 by removing from $\mathcal{C}^+$ edges whose both incident vertices are incident to some other edges of $\mathcal{C}^+$.

The resulting set $K$ is a spanning forest of $G$ composed of trees of depth 1. One vertex may thus be selected in each tree such that the tree rooted on this vertex has a depth 1. Selected vertices correspond to surviving vertices while remaning vertices are classified as non survivors.

The main motivation of this method is to provide a better stability of the reduction factor. The problem identified within the MIVS framework being connected to the vertex degree, the proposed method use the notion of maximal independent edge set to avoid the use of vertex's neighborhood. Experiments provided by Haximusa et al. [11] (Section 1.3.6) show that the reduction factor remains approximately equal to 2 along the pyramid.

### 1.3.5 Maximal independent directed edge set

Using the maximal independent edge set framework (Section 1.3.4), surviving vertices are selected so as to obtain rooted trees of depth one. Such a selection scheme is thus induced by the decomposition of the initial graph into trees and can not readily incorporate a priori constraints concerning surviving vertices. However, in some applications, the definition of the surviving vertex within each reduction window is an important issue. For example, within the analysis of line drawing framework [12], end points of a line or an intersection point must be preserved, and hence selected as survivors, for geometric accuracy reasons.

In order to provide a better selection of surviving vertices, Kropatsch et al [11] propose to adapt the MIES framework as follows: The decimation of a graph is based on the construction of a spanning forest $K$ such that:

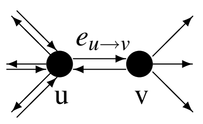
Cond. I : Any vertex of the graph belongs to exactly one tree of $K$ of depth one,

Cond. II : Each tree is encoded using directed edges. Its surviving vertex is defined either as the only vertex of the tree or as the unique target of all tree's edges.

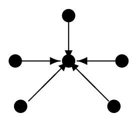
Each tree of $K$ defining a reduction window, Cond. I insures that the set of reduction windows encodes a partition of the vertex set of the graph. Moreover, the surviving vertex within each tree is uniquely determined by Cond. II. Such a use of directed edges, provides a simple way to constraint the selection of surviving vertices by restricting the construction of the forest $K$ to edges encoding possible selection of surviving and non surviving vertices. Such edges are called *preselected* edges.

Note that any undirected graph may be converted into a directed graph by transforming each of its undirected edge $e_{u,v}$ into a pair of directed reverse edges $e_{u\to v}$ and $e_{v\to u}$ with opposite sources and targets. However, the set of preselected edges of such a graph may contain $e_{u\to v}$ without containing $e_{v\to u}$.

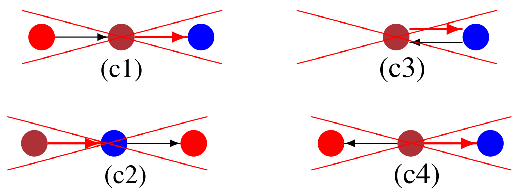
Let us denote by $G=(V,E)$ the directed sub graph induced by the set of preselected edges $E$. The first step toward the construction of the spanning forest $K$ consists to build a maximal independent set on $E$ using the following directed edge's neighborhood relationship (Fig. 1.3.5(a)):



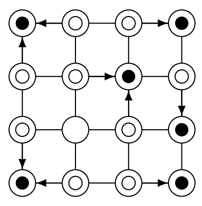
(a) $\mathcal{N}(e_{u\to v})$



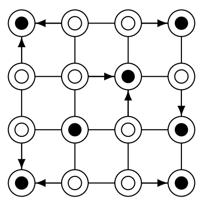
(b) A tree of a MIDES



(c) Forbiden configurations



(d) MIDES



(e)

Figure 1.11: A MIDES built on a $4 \times 4$ grid.

**Definition 2**

*Let $e_{u\to v}$ be a directed edge of $G$ with $u \neq v$. Then the directed neighborhood $\mathcal{N}(e_{u\to v})$ is given by all directed edges with the same source $u$, targeting the source $u$ or emanating from the target $v$:*

$$\mathcal{N}(e_{u\to v}) = \{e_{u\to w} \in E\} \cup \{e_{w\to u} \in E\} \cup \{e_{v\to w} \in E\}$$

Such a maximal independent set on the set of directed edges is called a *maximal independent directed edge set*(MIDES). Note that, since two neighbors can not be simultaneously selected within a maximal independent set, the definition of a neighborhood allows to specify forbidden configurations. Within the MIDES, once an edge $e_{u\to v}$ is selected, the set of edges incident to $u$ or $v$ which may be added to the MIDES is restricted to edges whose target is equal to $v$(Fig. 1.3.5(b)). A MIDES defines thus a forest of $G$ whose trees are composed edges pointing on a same target node. This unique target node is selected as a surviving vertex while source nodes of each tree are selected as non surviving vertices attached to this unique target. Fig. 1.3.5(c) represents some configurations of adjacent edges according to our neighborhood relationship (Def. 2). Such configurations cannot occur within a MIDES.

All trees of the forest defined by a MIDES have depth at most one and the surviving vertex of each tree is uniquely defined by the selected directed edges. Fig. 1.11 shows the results of a MIDES applied on a graph encoding a $4 \times 4$, 4-connected grid. Surviving vertices are represented with an extra filled circle (◉) while non surviving vertices are represented by two concentric circles (◎). As illustrated by this last figure, the resulting forest may not span all the vertices of the graph (vertex ◯, on the third row, second column). Such isolated vertices may be interpreted as trees of depth 0 and inserted into the forest in order to obtain a spanning forest $K$ fulfilling Cond. I and Cond. II. The construction of this spanning forest is thus

| Algorithms | | height | $\lvert\mathcal{V}_l\rvert / \lvert\mathcal{V}_{l+1}\rvert$ | | correction | |
|---|---|---|---|---|---|---|
| | | | $\mu_{pyr}$ | $\sigma_{pyr}$ | $\mu_{pyr}$ | $\sigma_{pyr}$ |
| MIVS | $\mu_{data}$ | 20.8 | 2.0 | 1.3 | 3.0 | 0.8 |
| | $\sigma_{data}$ | 5.2 | 0.3 | 1.1 | 0.2 | 0.1 |
| MIES | $\mu_{data}$ | 14.0 | 2.3 | 0.2 | 4.0 | 1.2 |
| | $\sigma_{data}$ | 0.14 | 0.01 | 0.05 | 0.1 | 0.1 |
| MIDES | $\mu_{data}$ | 12.0 | 2.6 | 0.3 | 2.8 | 1.1 |
| | $\sigma_{data}$ | 0.4 | 0.1 | 0.2 | 0.1 | 0.1 |

Table 1.1: Statistics on height of the pyramid, decimation ratio and iterations for correction.

achieved by the following three steps:

1. Define a maximal independent edge set on the directed graph $G = (V, E)$ where $E$ encodes a set of preselected edges.

2. Define the target of each tree of the resulting forest as a survivor and the sources as non surviving vertices attached to this unique target.

3. complete this forest with isolated vertices.

Note that contrary to the MIES (Section **??**), the MIDES construction scheme does not requires to split existing trees in order to obtain trees of depth one. Kropatsch et al. [11] schown that the decimation ratio obtain by a MIDES is at least 2.0 if the MIDES does not produce isolated vertices. Experiments performed by Kropatsch et al. in the same paper show that such isolated vertices does not occur frequently. Moreover, each isolated vertex needs only one tree of more edge to keep the balance for a reduction factor of 2.0.

### 1.3.6 Comparison of MIVS, MIES and MIDES

Kropatsch et al. [11] have compared the MIVS, MIES and MIDES construction schemes of a pyramid through several experiments based on a database of pyramids computed on initial graphs encoding $100 \times 100$, $150 \times 150$ and $200 \times 200$, 4-connected planar grids. The outcome of the random variable used within the maximal independent set is repeated with different seeds in order to obtain up to 1000 different pyramids. Statistics of all pyramids built by each specific selection ( MIVS, MIES and MIDES) are calculated to compare the properties of the different strategies by using the following parameters:

- The heigth of the pyramid (*height*),

- The reduction factor for vertices $\left(\frac{\lvert\mathcal{V}_l\rvert}{\lvert\mathcal{V}_{l+1}\rvert}\right)$,

- The number of iterations for correction required to complete a maximal independent set using the method based on the outcome of a random variable (Section 1.3.2) (*correction*).

Note that the reduction factor and the correction varry within a pyramid. We thus compute the mean ($\mu_{pyr}$) and standard deviation ($\sigma_{pyr}$) of these values for each pyramid and further compute the mean ($\mu_{data}$) and standard deviation ($\sigma_{data}$) of these global value over our whole dataset of pyramids. The resulting values are displayed in Table 1.1.

As shown by Table 1.1, MIVS provide a mean reduction factor equal to 2.01 but with an important standard deviation within the pyramid (1.3). This important variability of the decimation factor is coherent with the large mean height (20.8) of the pyramid. The maximal height on the dataset obtained using the MIVS being equal to 41. The mean number of iterations for correction is stable ($\sigma_{pyr} = 0.8$) and equal to 3.0. In conclusion, MIVS provide a mean reduction factor of 2.0 but such a mean result hide bad behaviors of the MIVS which have large chances to occur.

Compared to the MIVS, the MIES provides a larger mean decimation ratio (2.3), this decimation ratio being stable along the pyramid ($\sigma_{pyr} = 0.2$). The mean height of the pyramid is also lower (14.0) with a much lower standard deviation (0.14). The maximal height of a pyramid obtained using the MIES construction scheme within these experiments is equal to 15. The mean number of iterations for corrections (4.0) is however greater than the one of MIVS (3.0) with also a large standard deviation (1.2). In conclusions, MIES provide a more stable larger decimation ratio than MIVS and was greater than its theorical bound in all experiments.

The mean reduction factor obtained using the MIDES construction scheme (2.6) is the largest of the three methods with a low variation within pyramids ($\sigma_{pyr} = 0.3$). The mean height of the pyramid (12.0) is also lower than the obtained with MIES and MIVS and remains stable across the database ($\sigma_{data} = 0.4$). The number of iterations needed to complete the maximal independent set is comparable with the one of MIVS. The MIDES provides thus both a large and stable reduction factor and a low number of iterations for correction.

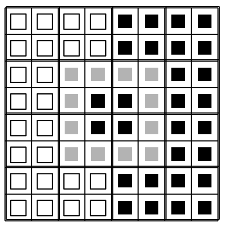
## 1.4 Irregular Pyramids and Image properties

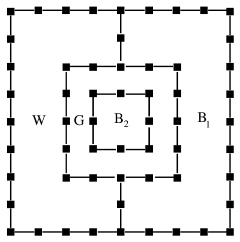
Different familly of irregular pyramids have been proposed [5, 13, 14] to encode image content. These pyramids may vary both according to the method used to build one graph of the pyramid from the graph bellow and according to the type of graph used to encode each level. The construction scheme of a pyramid, described in Section 1.3, determines its decimation ratio and may be understood as the pyramid's vertical dynamic. Conversely, the choice of a given graph model determines which set of topological and geometrical image properties may be encoded at each level. This last choice may be understood as the determination of the horizontal resolution of the pyramid. Let us additionally note that choices concerning pyramid's construction scheme and graph encoding are usually independent since different pyramid construction schemes may be applied using different graph encodings. The main geometrical and topological properties of an image together with the different graph models used to encode them within the irregular pyramid framework are detailed in the following sections.

### 1.4.1 Relating Regions

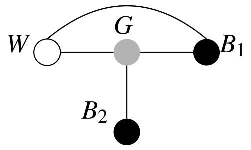
Within an image analysis scheme, high level image description is usually based on low level image processing steps. Segmentation stage is one of this key step. Segmentation stage produces an image's partition into meaningful connected set of pixels called *regions*. However, in many applications, low level image segmentation stage cannot be readily separated from high level gloals. On the contrary, segmentation algorithms should often extract fine information about a partition in order to guide the segmentation process according to high level objectives. There is thus a need to design graph models which can be both efficiently updated during a segmentation stage and which allow to extract fine information about the partition.

Region's adjacency also called *meet relationship* consitutes a basic but widely used relationship between regions. Two regions of a partition are said to be adjacent if they share at least one common boundary. Fig. 1.12(a) shows a partition of an $8 \times 8$ image whose boundaries are displayed in Fig. 1.12(b) using inter-pixels elements such as lignels (or cracks) and pointels [15, 16]. One may see from these figures that meet relationships corresponds to different configurations of regions. For example, the central gray region $G$ contains a black region $B_2$ and shares one common boundary with an other black region $B_1$. In the same way, white and black regions, $W$ on the left and $B_1$ on the right part of the image respectively, share two connected boundaries.


(a) Initial image



(b) Inter pixels



(c) Simple graph

Figure 1.12: An image partition (a) whose boundaries are encoded by inter pixel elements(b). A Simple graph encoding of the resulting partition(c).

Finer relationships between regions such as RCC-8 defined by Randel [17] or the relationships defined by K. Shearer et al. [18] in the context of graph matching should thus be used in order to describe fine relationships between regions. Within the particular context of image segmentation, the following relationships may be defined from these two models:

1. The different models used to encode partitions either encode the existence of at least one common boundary between two region or create one relationship for each boundary between these regions. We denote these two types of encodings *meets_exists* and *meets_each*. A model encoding only the meets_exists relationship is thus not able to distinguish a simple from a multiple adjacency relationship between two regions such as the one between vertices $W$ and $B_1$ in Fig. 1.12. Note that, the ability of models to retrieve efficiently a given common boundary between two regions is also an important feature of these models.

2. The relationship *A contains B* expresses the fact that region B is inside region A. For example, gray region in Fig. 1.12(a) contains a black region.

3. The *inside* relationship is the inverse of the *contains* relation: A region *B inside A* is contained in A.

One additional relationship not directly handled by Shearer and Randel's models may be defined within the hierarchical segmentation scheme. Indeed within such a framework a region defined at a given level of a hierarchy is *composed of* regions defined at levels below.

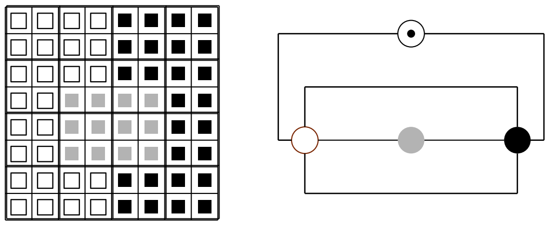
The following relationships may thus be deduced from the relationships defined by Shearer and Randel and the example provided by Fig. 1.12(a) : The *meets_exists, meets_each, contains, inside* and *composed of.* Note that unlike meets relationships, the contains and inside relations are asymmetric. A contains or inside relation between two regions allows thus to characterize each of the regions sharing this relation.

### 1.4.2 Graph models

Different graph models [19] have been proposed to describe the content of an image within the image analysis framework. Each of this graph model encodes a different subset of the relationships defined in Section 1.4.1. We present in the following sections, the main graph models introduced within the irregular pyramid framework together with a description of the relationships between regions encoded by such models.

#### Simple Graph

One of the most common graph data structure, within the segmentation framework is the *Region Adjacency Graph* (RAG). A RAG is defined from a partition by associating one vertex to each region and by creating an edge between two vertices if the associated regions share a common boundary (Fig. 1.12(c)). A RAG encodes thus the meet_exists relationship. It corresponds to a simple graph without any double edge between


(a) $G = (V,E)$

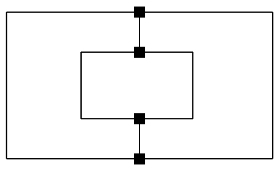

(b) $\overline{G} = (\overline{V},\overline{E})$

Figure 1.13: A non simple graph encoding a partition (a) and its dual (b).

vertices nor self-loop. Within a non-hierarchical segmentation scheme the RAG model is usually applied as a merging step to overcome the over-segmentation produced by the previous splitting algorithm [20]. Indeed, the existence of an edge between two vertices denotes the existence of at least one common boundary segment between the two associated regions which may thus be merged by removing this segment. Within this framework, edge information may thus be interpreted as a possibility to merge two regions identified by the vertices incident to the edge. Such a merge operation implies to collapse the two vertices incident to the edge into one vertex and to remove this edge. This operation is called an *edge contraction*. In order to obtain a simple graph, we should further remove any self loop or double edge incident to the newly created vertex that may have been created by the contraction operation.

The contraction operation is defined for any edge not defining a loop and thus for any set of edges which does not contain a cycle (i.e. a forest). Given a graph $G = (V,E)$ and a set $N \subset E$ defining a forest of $G$, the contraction of $N$ in $G$ is denoted $G/N$. In the same way, given a set $M$ of edges, the removal of $M$ in $G$ is denoted $G \setminus M$.

The RAG model encodes thus only the existence of a common edge between two regions (the *meets_exists* relationship). Moreover, the existence of a common edge between two vertices does not provide enough information to differentiate a meets relationship from a *contains* or *inside* one. This drawback is illustrated in Fig. 1.12(c) where the relationships between $G$ and $B_1$ is the same than the one between $G$ and $B_2$ (both $B_1$ and $B_2$ are adjacent to $G$). However, the gray region associated to $G$ contains the black region associated to $B_2$ but just shares one connected boundary with the black region encoded by $B_1$. These two different configurations of regions are not encoded by the simple graph model.

## Dual Graph Model

Edges within a RAG model only encode the meet_exists relationship. One straightforward solution to encode multiple boundaries between regions consists to define non simple graphs by creating one edge between two vertices for each common boundary between two regions. Fig. 1.13(a) shows such an encoding where the two connected boundaries between black and white regions are encoded by two edges between the corresponding vertices. Vertex (⊙) encodes the background of the image. White and black regions being adjacent to the background, vertices encoding these regions are adjacent to the vertex encoding the background. Such a graph encoding does not allow to perform readily graph simplifications. This last drawback is illustrated in Fig. 1.14(b) whose graph is obtained by contracting the edge between dark gray (●) and white (○) vertices defined in Fig. 1.14(a). This edge contraction operation induces the merge of the corresponding dark gray and white regions in Fig. 1.14(a) hence producing a partition similar to the one displayed in Fig. 1.13(a). As illustrated in Fig. 1.14(b), edge contraction between dark gray and white vertices induce the creation of two edges between white and gray vertices. These two edges encode a single boundary between the white and gray regions and should thus be merged into a single one. However, the use of a single graph does not allow to readily distinguish these two edges from the multiple edges between white and black vertices which encode two non connected boundaries and should thus be preserved by any simplification step.

The dual graph model proposed by Kropatsch [21] allows to perform efficient simplifications of non simple graphs after a set of edge contractions. This model assumes that objects to be modeled lie in the 2D plane and may thus be encoded by a planar graph. Using this assumption, the dual graph model is defined as a couple of dual graphs $(G, \overline{G})$, where $\overline{G}$ is the dual of $G = (V, E)$. The set of vertices and edges of $G$ is defined as follows (Fig. 1.13(a)):

- Each vertex of $V$ encodes a connected component of the image, one additional special vertex encodes the background of the image,
- Each edge of $E$ encodes a connected boundary between two connected components.

The dual graph of $G$, $\overline{G} = (\overline{V}, \overline{E})$ is defined by creating one vertex (■, Fig. 1.13(b)) of $\overline{G}$ for each face of $G$ and connecting these vertices such that each edge of $G$ is crossed by one edge of $\overline{G}$. Note that $G$ may encode several edges between two vertices. It is thus a non simple graph. Its dual $\overline{G} = (\overline{V}, \overline{E})$ being also non simple.

Each dual graph $\overline{G}$ being deduced automatically from the primal graph $G$, both graphs $G$ and $\overline{G}$ share important properties, which are recalled below:

- The dual graph operation is idempotent:$\overline{\overline{G}} = G$. This important property means that the dual operation does not induce any loss of infomation since the primal graph may be recovered from its dual. On graph and its dual encode thus a same information differently.
- Since each edge of $\overline{G}$ crosses one edge of $G$, there is a one to one correspondance between edges of $G$ and $\overline{G}$. Given a set $N$ of edges, we will denote by $\overline{N}$ the corresponding set of edges in $\overline{G}$.
- The contraction of any forest $N$ in the primal graph is equivalent to the removal of $\overline{N}$ in its dual. In other words:

$$\overline{G/N} = \overline{G} \setminus \overline{N} \tag{1.9}$$

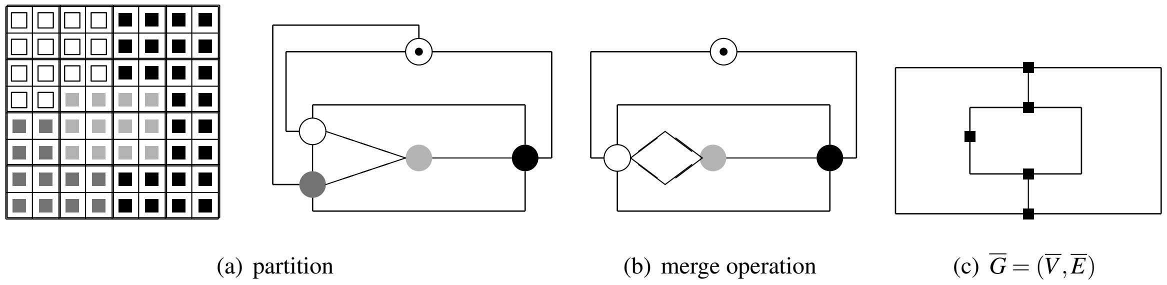

(a) partition (b) merge operation (c) $\overline{G} = (\overline{V}, \overline{E})$

Figure 1.14: A partition into 4 regions and its associated non simple graph $G$(a), the graph $G$ obtained by the contration of the edge between white and dark gray vertices. The dual graph of (b), $\overline{G} = (\overline{V}, \overline{E})$ (c).

- Application of the above equation to the dual graph $\overline{G}$ instead of $G$ leads to the following equation:

$$\overline{G}/\overline{N} = \overline{G \backslash N} \tag{1.10}$$

In other words, the contraction of any forest $\overline{N}$ of $\overline{G}$ is equivalent to the removal of $N$ in $G$.

As illustrated by Fig. 1.14(b), edge contraction may induce the creation of some redundant edges. These edges belong to one of the following categories:

- Redundant double edge: These edges encode multiple adjacency relationships between two vertices and define degree two faces. They can thus be characterized in the dual graph as degree two dual vertices (Fig. 1.14( b and c)). In terms of partition's encoding these edges correspond to an artificial split of one boundary segment between two regions.

- Empty self-loop: These edges correspond to a self-loop with an empty inside. These edges define thus degree one faces and are characterized in the dual graph as degree one vertices. Such edges encode artificial inner boundaries of regions.

Both double edges and empty self-loops do not encode relevant topological relations and can be removed without any harm to the involved topology [21]. The resulting pair of dual graph encodes each connected boundary segment between two regions by one edge between corresponding vertices. The Dual Graph model encodes thus the *meets_each* relationship through multiple edges between vertices.

Within the dual graph model, the encoding of the adjacency between two regions one inside the other is encoded by two edges (Fig. 1.15): One edge encoding the common border between the two regions and one self-loop incident to the vertex encoding the surrounding region. One may think to characterize the inside relationship by the fact that the vertex associated to the inside region should be surrounded by the self-loop. However, as shown by Fig. 1.15(c) one may exchange the surrounded vertex without modifying the incidence relationships between both vertices and faces. Two dual graphs being defined by these incidence

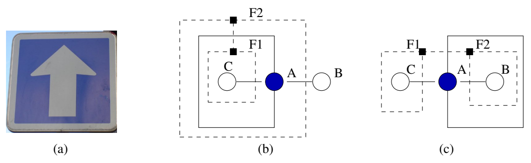
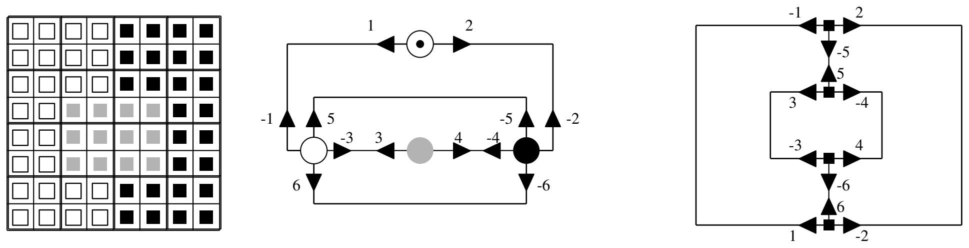


Figure 1.15: The Graph (b) defines the top of a dual graph pyramid encoding an ideal segmentation of (a). The self loop incident to vertex A may surround either vertex B or C without changing the incidence relations between vertices and faces. The dual vertices associated to faces are represented by filled boxes (■). Dual edges are represented by dashed lines.



(a) a partition (b) $G = (\mathcal{D}, \sigma, \alpha)$ (c) $\overline{G} = (\mathcal{D}, \varphi = \sigma \circ \alpha, \alpha)$

Figure 1.16: A partition (a) encoded by a combinatorial map (b) and its dual (c).

relationships one can exchange the surrounded vertex without modifying the encoding of the graphs. This last remark shows that the *inside/contains* relationships cannot be characterized locally within the dual graph framework.

## Combinatorial maps

Combinatorial maps and generalized combinatorial maps define a general framework which allows to encode any subdivision of nD topological spaces orientable or non-orientable with or without boundaries. An exhaustive comparison of combinatorial maps with other boundary representations such as cell-tuples and quad-edges is presented in [22]. Recent trends in combinatorial maps apply this framework to the segmentation of 3D images [23] and the encoding of hierarchies [24, 25].

This chapter is devoted to $2D$ combinatorial map which we simply call combinatorial maps. A combinatorial map may be seen as a planar graph encoding explicitly the orientation of edges around a given vertex. Figure 1.16 demonstrates the derivation of a combinatorial map from a plane graph $G = (V, E)$ (Fig. 1.13(a)). First the edges of $E$ are split into two half edges called *darts*, each dart having its origin at the vertex it is attached to. The fact that two half-edges (darts) stem from the same edge is recorded in the reverse permutation $\alpha$. A second permutation $\sigma$ encodes the set of darts encountered when turning counter-clockwise around a vertex. A combinatorial map is thus defined by a triplet $G = (\mathcal{D}, \sigma, \alpha)$, where $\mathcal{D}$ is the set of darts and $\sigma$, $\alpha$ are two permutations defined on $\mathcal{D}$ such that $\alpha$ is an involution:

$$\forall d \in \mathcal{D} \quad \alpha^2(d) = d \tag{1.11}$$

If darts are encoded by positive and negative integers, involution $\alpha$ may be implicitly encoded by the sign (Figure 1.16(a)).

Given a dart $d$ and a permutation $\pi$, the $\pi$-cycle of $d$ denoted by $\pi^*(d)$ is the series of darts $(\pi^i(d))_{i\in\mathbb{N}}$ defined by the successive applications of $\pi$ on the dart $d$. The $\sigma$ and $\alpha$ cycles of a dart $d$ will be respectively denoted by $\sigma^*(d)$ and $\alpha^*(d)$.

The $\sigma$-cycle of one vertex encodes its adjacency relationships with neighboring vertices. For example, let us consider the dart 5 in Fig. 1.16(a) attached to the white vertex. Turning counterclockwise around this vertex we encounter the darts $5, -1, 6$ and $-3$. We have thus $\sigma(5) = -1$, $\sigma(-1) = 6$, $\sigma(6) = -3$ and $\sigma(-3) = 5$. The $\sigma$-cycle of 5 is thus defined as $\sigma^*(5) = (5, -1, 6, -3)$. The whole permutation $\sigma$ is defined as the composition of its different cycles and is equal to $\sigma = (5, -1, 6, -3)(4, 3)(-5, -4, -6, -2)(1, 2)$. Permutation $\alpha$ being implicitly encoded by the sign in Fig. 1.16, we have $\alpha^*(5) = (5, -5)$ and $\alpha = (1, -1)(2, -2)\dots(6, -6)$.

We can already state one major difference between a combinatorial map and an usual graph encoding of a partition. Indeed, a combinatorial map may be seen as a planar graph with a set of vertices (the cycles of $\sigma$) connected by edges (the cycles of $\alpha$). However, compared to an usual graph encoding, a combinatorial map encodes additionally the local orientation of edges around each vertex thanks to the order defined within each cycle of $\sigma$.

Given a combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$, its dual is defined by $\overline{G} = (\mathcal{D}, \varphi, \alpha)$ with $\varphi = \sigma \circ \alpha$. The cycles of the permutation $\varphi$ encode the sequence of darts encountered when turning around a face of $G$. Note that, using a counter-clockwise orientation for permutation $\sigma$, each dart of a $\varphi$-orbit has its associated face on its right (see e.g. the $\varphi$-orbit $\varphi^*(5) = (5, -4, 3)$ in Figure 1.16(b)). Cycles of $\varphi$ may alternatively be interpreted as the sequence of darts encountered when turning clockwise aroud each dual vertex (Figure 1.16(c)). Permutation $\varphi$, on this last figure, being defined as $\varphi = (-1, 2, -5)(5, -4, 3)(4, -6, -3)(1, 6, -2)$

The combinatorial map formalism allows to encode each connected boundary between two regions by one edge. Models based on combinatorial maps encode thus the *meets_each* relationship. However, within the combinatorial map framework two connected components $S$ inside $R$ of a partition will be encoded by two combinatorial maps without any information about the respective positioning of $S$ and $R$. Models based on combinatorial maps have thus designed additional data structure like the inclusion tree [26] or the Parent-Child relationships [27, 16] to encode the *contains* and *inside* relationships. Using these models any modification of the partition implies to update both the combinatorial maps and the additional data structures.

### 1.4.3 Simple Graph Pyramids

Simple graphs pyramids are defined as a stack of successively reduced simple graphs, each graph being built from the graph below by selecting a set of vertices named surviving vertices and mapping each non-surviving vertex to a surviving vertex [6, 5]. Using such a framework, the graph $G_{l+1} = (V_{l+1}, E_{l+1})$ defined at level $l+1$ is deduced from the graph defined at level $l$ by the following steps:

1. Define a partition of $V_l$ into connected reduction windows and select one surviving vertex within each reduction window. Surviving vertices define the set $V_{l+1}$ while non surviving vertices of each reduction windows are adjacent to the unique survivor of the reduction window (Section 1.3).

2. Define the adjacency relationships between the vertices of $G_{l+1}$ in order to define $E_{l+1}$.

The selection of surviving vertices and the definition of a partition of $V_l$ into reduction windows is described in Section 1.3.2. Note that the attachement of each non surviving vertex to the unique survivor of its reduction window induces a father/child relationship between $V_l$ and $V_{l+1}$. Each non surviving vertex is the child of the unique survivor of its reduction window. Conversely, any surviving vertex is the father of all non surviving vertices within its reduction window.

We are thus here, mainly concerned with the second and last step of the decimation process which consists to connect surviving vertices in $G_{l+1}$ in order to define $E_{l+1}$. Meer [6] attaches the outcome of a random variable to each vertex and defines a partition of $V_l$ into reduction windows using the Maximal Independent Vertex Set (MIVS) construction scheme (Section 1.3.2). Fig. 1.17(a) (see also Fig. 1.9) shows a decomposition of a $4 \times 4$, 8-connected grid into 4 reduction windows using such a construction scheme. Surviving vertices are represented by an extra circle and reduction windows are superimposed to the figure. Given this decomposition of $V_l$ into reduction windows, Meer joins two fathers by an edge if they have adjacent children (e.g. vertices labeled 15 and 7 in Fig. 1.17(a)). Since each child is attached directly to its father, two adjacent fathers at the reduced level are connected in the level below by paths of length less than three. Let us call these paths *connecting paths*. Using Constraint (1.4) of a MIVS, two surviving vertices cannot be adjacent in $G_l$. Therefore, the length of the connecting paths is between 2 and 3. Using the restriction of the decimation process to subgraphs defined by Montanvert and Jolion [5, 7] two surviving vertices may be adjacent hence leading to connecting path of length 1. The set of edges $E_{l+1}$ induced by connecting paths between surviving vertices in Fig. 1.17(a) is shown in Fig. 1.17(b).

Two surviving vertices are thus connected in $G_{l+1}$ if they are connected in $G_l$ by a path of length lower or equal than 3. Two reduction windows adjacent by more than one such path will thus be connected by a single edge in the reduced graph. The stack of graphs produced by the above decimation process is thus a stack of simple graphs each simple graph encoding only the existence of one common boundary between

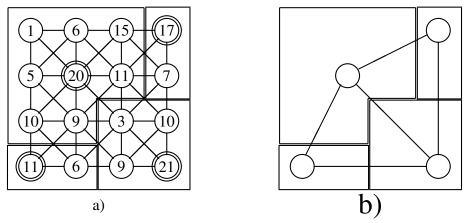


Figure 1.17: The reduced graph deduced from the maximal independent vertex set (a) and the Parent-Children relationships (b).

two regions (the *meeets_exists* relationship). Moreover, as mentioned in Section 1.4.2 the RAG model which corresponds to a simple graph does not allow to encode *contains* and *inside* relationships.

Let us finally note that the above construction scheme of the reduced graph $G_{l+1} = (V_{l+1}, E_{l+1})$ from $G_l = (V_l, E_l)$ may be adapted to all parallel reduction schemes detailed in Section 1.3. Using different reduction schemes than the MIVS, it may be interesting to note that the construction of $G_{l+1}$ from $G_l$ may be equivalently performed by merging all the vertices of each reduction window into a single vertex (Section 1.4.2). This operation may be decomposed as follows:

1. The selection of an unique edge between each non surviving vertex and its father.
2. The contraction of the set of edges previously selected,
3. The removal of any double edge or self-loop that may have been created by the previous step.

Note, that MIES (Section 1.3.4) and MIDES (Section 1.3.5) construction schemes explicitely connect each non surviving vertex to its father hence providing the first step of the above process.

### 1.4.4 Dual Graph Pyramids

As mentioned in Section 1.4.3 the construction of each level of a simple graph pyramid may be decomposed into a contraction step, followed by the removal of any multiple edge or self-loop. Since each edge to be contracted connects a non surviving vertex to its surviving father, the set $K$ of contracted edges defines a forest of $K$.

#### Contraction Kernel

The different parallel construction schemes of irregular pyramids presented in Section 1.3 allows to define different forests $K$ composed of trees of depth at most one hence allowing an efficient parallel contraction of each reduction window.

Kropatsch [3] propose to abstract the set $K$ of edges to be contracted through the notion of *Contraction Kernel*. A Contraction Kernel is defined on a graph $G = (V, E)$ by a set of surviving vertices $S$, and a set of non surviving edges $K$ (represented by bold lines in Fig. 1.18) such that:

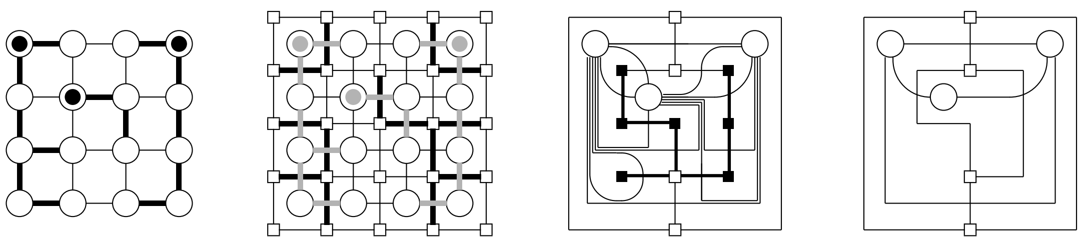
- $K$ is a spanning forest of $G$,
- Each tree of $K$ is rooted by a vertex of $S$.

Since the set of non surviving edges $K$ forms a forest of the initial graph, no self loop may be contracted and the contraction operation is well defined. The decimation of a graph by Contraction Kernels differs from the one defined in Section 1.4.3 on the two following points:

- First, using Contraction Kernels, the set of surviving vertices is not required to form a MIVS. Therefore, two surviving vertices may be adjacent in the contracted graph.
- Secondly, using Contraction Kernels a non surviving vertex is not required to be directly linked to its father but may be connected to it by a branch of a tree (see Fig. 1.18(a)). The set of children of a surviving vertex may thus vary from a single vertex to a tree with any height.

Let us note that the reduction of a graph by a sequence of contraction kernels $K_1, \dots, K_n$ composed of trees of depth one is equivalent to the application of a single contraction kernel $K$ on the initital graph [28]. The set of edges of $K$ is defined as the union of all edges of the contraction kernels $(K_i)_{i \in \{1,\dots,n\}}$. Conversely, any contraction kernel may be decomposed into a sequence of contractions kernels composed of trees of depth ones. Contraction kernels should thus be understood as an abstraction of the forest of trees of depth one defined in Section 1.3

Fig. 1.18(a) illustrates a contraction kernel $K$ composed of 3 trees represented by thick lines ( ▬ ). The root of each tree being represented by dot circles (◉).



(a) $G = (V, E)$ and $K$ (b) $\overline{G} = (\overline{V}, \overline{E})$ and $\overline{K}$ (c) $G' = G/K(\circ)$, $\overline{G'} = \overline{G} \setminus \overline{K}(\square)$ and $\overline{K'}$ (d) $G' \setminus K'$ and $\overline{G}/\overline{K}$

Figure 1.18: A contraction Kernel $(S, N)$ composed of three trees. Vertices belonging to $S$ are represented with a filled circle inside(●). Dual vertices incident to redundant dual edges are represented by filled boxes.

This kernel is defined on a graph $G$ encoding a $4 \times 4$ planar sampling grid. The dual of $G$ is shown in Fig. 1.18(b) together with the set of dual edges $\overline{K}$. The graph $G$ is superimposed to $\overline{G}$ in order to highlight relationships between $K$ and $\overline{K}$. Contraction of $K$ in $G$ is equivalent to the removal of $\overline{K}$ in $\overline{G}$ (equation 1.9, Section 1.4.2). The resulting pair of reduced graph $(G/K, \overline{G} \setminus \overline{K})$ is shown in Fig. 1.18(c).

As mentioned in Section 1.4.2, the contraction of a set of edges may create redundant edges such as double edges surrounding degree 2 faces and self loops corresponding to degree one faces. These redundant edges may be thus efficiently characterized in the dual graph respectively as degree 2 and degree 1 dual vertices. Such redundant dual vertices are shown in Fig. 1.18(c) as filled rectangles. Removal of degree one dual vertices may be achieved by contracting the only edge incident to such vertices. In the same way, degree 2 dual vertices may be removed by contracting one of their incident edge. The resulting set of edges $\overline{K'}$ defines a forest of the dual graph $\overline{G}$ and is called a *removal kernel* (thick edges (— ) in Fig. 1.18(c)). Indeed, contraction of $\overline{K'}$ in $\overline{G}$ is equivalent to the removal of $K'$ in $G$ (Section 1.4.2). The reduction of a dual graph $(G, \overline{G})$ is thus performed in two steps:

- A first contraction step which computes $(G', \overline{G'}) = (G/K, \overline{G} \setminus K)$ from $(G, \overline{G})$ (Fig. 1.18(a-c)).
- The definition of a removal kernel $\overline{K'}$ from $(G', \overline{G'})$ which removes all double edges and empty self loops of $G$ by computing $(G' \setminus K', \overline{G'}/K')$ (Fig. 1.18(d)).

A pyramid of dual graphs introduced by Kropatsch [3] is defined as a stack of pairs of dual graphs $((G_0, \overline{G_0}), \ldots, (G_n, \overline{G_n}))$, successively reduced. Each graph $G_{i+1}$ is deduced from $G_i$ by a contraction kernel $K_i$ followed by the application of a removal kernel removing any redundant double edge or empty self loop which may have been created by the contraction step. The initial graph $G_0$ and its dual $\overline{G_0}$ may encode a planar sampling grid or be deduced from an over segmentation of an image.

**Dual Graph pyramids and Topological relationships**

Given one tree of a contraction kernel, the contraction of its edges collapses all the vertices of the tree into a single vertex and keeps all the connections between the vertices of the tree and the remaining vertices of the graph. The multiple boundaries between the newly created vertex and the remaining vertices of the graph are thus preserved. Each graph of a dual graph pyramid encodes thus the *meets_each* relationships. This property is not modified by the application of a removal kernel which only removes redundant edges.

Moreover, due to the forest requirement, the encoding of the adjacency between two regions one inside the other will be encoded by two edges (Fig. 1.15): One edge encoding the common border between the two regions and one self-loop incident to the vertex encoding the surrounding region. As mentioned in Section 1.4.2 (Fig. 1.15), such a couple of edges does not allow to characterize locally contains and inside relationships.

### 1.4.5 Combinatorial Pyramids

As in the dual graph pyramid scheme [29] (Section 1.4.4) a combinatorial pyramid is defined by an initial combinatorial map successively reduced by a sequence of contraction or removal operations. The initial combinatorial map encodes a planar sampling grid or a first segmentation and the remaining combinatorial maps of a combinatorial pyramid encode a stack of image partitions successively reduced. As mentioned in Section 1.4.2, page 25 a combinatorial map may be understood as a dual graph with an explicit encoding of the orientation of the edges incident to each vertex. This explicit encoding of the orientation is preserved within the combinatorial pyramid using contraction and removal operations equivalent to the operations used for dual graphs but which preserve the orientations of edges around the vertices of the reduced combinatorial maps [30, 31].

Contraction operations are controlled by contraction kernels (CK) defined as a set of darts encoding the set of edges to be contracted. Given a combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$ a kernel $K$ defined on $G$ is thus included in $\mathcal{D}$ and symmetric according to $\alpha$:

$$\forall d \in K \ \alpha(d) \in K$$

The removal of redundant edges is performed as in the dual graph reduction scheme by a removal kernel. Such kernels are, as contraction kernels, defined as set of darts to be deleted. Contrary to the dual graph pyramid framewok, removal kernel are decomposed in two sub-kernels: A removal kernel of empty self-loops (RKESL) which contains all darts incident to a degree 1 dual vertex and a removal kernel of empty double edges (RKEDE) which contains all darts incident to a degree 2 dual vertex. These two removal kernels are defined as follows: The removal kernel of empty self-loops RKESL is initialized by all self-loops surrounding a dual vertex of degree 1. RKESL is further expanded by all self-loops that contain only other self-loops already in RKESL until no further expansion is possible. For the removal of empty double edges RKEDE we ignore all empty self-loops in RKESL in computing the degree of the dual vertex. Note that the successive application of a RKESL and a RKEDE is equivalent to the application of a removal kernel defined within the dual graph framework. However, this decomposition allows to distinguish darts encoding boundaries between two regions from inner boudaries within the implicit encoding scheme of combinatorial pyramids (next section). Both contraction and removal operations defined within the combinatorial pyramid framework are thus defined as is the dual graph framework but additionally preserve the orientation of edges around each vertex. Further details about the construction scheme of a Combinatorial Pyramid may be found in [30].

**Implicit encoding of a combinatorial pyramid**

Let us consider an initial combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$ and a sequence of kernels $K_1, \ldots, K_n$ successively applied on $G_0$ to build the pyramid. Each combinatorial map $G_i = (\mathcal{SD}_i, \sigma_i, \alpha_i)$ is defined from $G_{i-1} = (\mathcal{SD}_{i-1}, \sigma_{i-1}, \alpha_{i-1})$ by the application of the kernel $K_i$ on $G_{i-1}$ and the set of surviving darts at level $i$: $\mathcal{SD}_i$ is equal to $\mathcal{SD}_{i-1} \setminus K_i$. We have thus:

$$\mathcal{SD}_{n+1} \subset \mathcal{SD}_n \subset \ldots \mathcal{SD}_1 \subset \mathcal{D} \tag{1.12}$$

The set of darts of each reduced combinatorial map of a pyramid is thus included in the base level combinatorial map. This last property allows us to define the two following functions:

1. one function *state* from $\{1, \ldots, n\}$ to the states $\{CK, RKESL, RKEDE\}$ which specifies the type of each kernel.

2. One function *level* defined for all darts in $\mathcal{D}$ such that $level(d)$ is equal to the maximal level where $d$ survives:

   $$\forall d \in \mathcal{D}\ level(d) = \ Max\{i \in \{1, \ldots, n+1\} \,|\, d \in \mathcal{SD}_{i-1}\}$$

   a dart $d$ surviving up to the top level has thus a level equal to $n+1$. Note that if $d \in K_i, i \in \{1, \ldots, n\}$ then $level(d) = i$.

We have shown [32, 30] that the sequence of reduced combinatorial maps $G_0, \ldots, G_{n+1}$ may be encoded without any loss of information using only the base level combinatorial map $G_0$ and the two functions *level* and *state*. Such an encoding is called an *implicit encoding* of the pyramid.

The encoding of the function *state* being proportional to the height of the pyramid its memory cost is negligible compared to the cost of the base level combinatoiral map. The encoding of the function level requires to associate one integer to each dart. The memory cost of such a function for a pyramid composed of $n$ levels is thus equal to $\log_2(n)\,|\mathcal{D}|$. Using an implicit encoding of the involution $\alpha$, the encoding of permutation $\sigma$ requires $|\mathcal{D}|\log_2(|\mathcal{D}|$ bytes. The total memory cost of an implicit pyramid is thus equal to:

$$[\log_2(n) + \log_2(|\mathcal{D}|)]\,|\mathcal{D}|\,. \tag{1.13}$$

On the other hand, using an explicit encoding of a combinatorial pyramid, if we suppose that the number of darts decreases by a factor 2 between each level, the total number of darts of the whole pyramid is equal to:

$$|\mathcal{D}|\sum_{i=0}^{p}\frac{1}{2^i} \approx 2\,|\mathcal{D}|$$

where $|\mathcal{D}|$ is supposed to be a power of 2 ($|\mathcal{D}| = 2^p$). The total memory cost of the explicit encoding of a combinatorial pyramid is thus equal to:

$$2\,|\mathcal{D}|\log_2(|\mathcal{D}|). \tag{1.14}$$

The ratio between equations 1.13 and 1.14 is equal to:

$$\frac{1}{2}\left(1+\frac{\log_2(n)}{\log_2(|\mathcal{D}|)}\right)$$

$\log_2(n)$ being usually much lower than $\log_2(|\mathcal{D}|)$ the implicit encoding provides a lower memory cost than the explicit encoding. Let us additionally note that if we bound the maximal level of a pyramid, using for example 32 bits integers to store the function level, we obtain a hierarchical encoding which is *in practice* independent of the height of the pyramid.

## Dart's embedding and Segments

The receptive field of a dart $d \in \mathcal{SD}_i$ corresponds to the set of darts reduced to $d$ at level $i$ [30, 32]. Using the implicit encoding of a combinatorial pyramid, the receptive field $RF_i(d)$ of $d \in \mathcal{SD}_i$ is defined as a sequence $d_1.\dots.d_q$ of darts in $\mathcal{D}$ by $d_1 = d$, $d_2 = \sigma_0(d)$ and for each $j$ in $\{3,\dots,q\}$ :

$$d_j = \begin{cases} \varphi_0(d_{j-1}) & \text{if } state(level(d_{j-1})) = CK \\ \sigma_0(d_{j-1}) & \text{if } state(level(d_{j-1})) \in \{RKEDE, RKESL\} \end{cases} \tag{1.15}$$

The dart $d_q$ is defined as the last dart of the sequence which have been contracted or removed below the level $i$. Therefore, the successor of $d_q$ according to equation 1.15, $d_{q+1}$ satisfies $level(d_{q+1}) > i$. Moreover, we have shown [30, 32] that $d$, $d_q$ and $d_{q+1}$ are additionally connected by the two following relationships:

$$\sigma_i(d) = d_{q+1} \text{ and } \alpha_i(d) = \alpha_0(d_q) \tag{1.16}$$

Using usual graph encodings of image content, a region corresponds to a vertex and two regions are connected by an edge if the two regions share a boundary. In the Combinatorial Map, vertices and edges correspond to $\sigma$ and $\alpha$ cycles respectively. Therefore, each dart $d \in \mathcal{SD}_i$ encodes a boundary between the regions associated to $\sigma_i^*(d)$ and $\sigma_i^*(\alpha_i(d))$. Moreover, in the lower levels of the pyramid the two vertices of an edge may belong to a same region. We call the corresponding boundary segment an *internal boundary* in contrast to an *external boundary* which separates two different regions. The receptive field of $d$ at level $i$ ($RF_i(d)$) contains both darts corresponding to this boundary and additional darts corresponding to internal boundaries. The sequence of external boundary darts contained in $RF_i(d)$ is denoted by $\partial RF_i(d)$ and is called a *segment*. The order on $\partial RF_i(d)$ is deduced from the receptive field $RF_i(d)$. Given a dart $d \in \mathcal{SD}_i$, the sequence $\partial RF_i(d) = d_1,\dots,d_q$ is retrieved by [32]:

$$d_1 = d \text{ and } \forall j \in \{1,\dots,q-1\}\ d_{j+1} = \varphi_0^{n_j}(\alpha_0(d_j)) \tag{1.17}$$

The dart $d_q$ is the last dart of $\partial RF_i(d)$ which belongs to a double edge kernel. This dart is thus characterized using equation 1.16 by $d_q = \alpha_0(\alpha_i(d))$. Note that each dart of the base level corresponds to an oriented crack [15, 16] (Section 1.4.2). A segment corresponds thus to a sequence of oriented cracks encoding a connected boundary between two regions [32].

The value $n_j$ is defined for each $j \in \{1, \ldots, q-1\}$ by :

$$n_j = Min\{k \in \mathbb{N}^* \mid state(level(\varphi_0^k(\alpha_0(d_j)))) = RKEDE\}. \tag{1.18}$$

A segment may thus be interpreted as a maximal sequence, according to equation 1.17, of darts removed as double edges. Such a sequence connects two darts ($d$ and $\alpha_0(d_q) = \alpha_i(d)$) surviving up to level $i$. The retrieval of the boundaries using equations 1.17 and 1.18 is one of the major reason which lead us to distinguish empty self-loop removal kernels and double edges.

Let us additionally note that if $G_0$ encodes the 4-connected planar sampling grid, each $\varphi_0$ cycle is composed of at most 4 darts (Fig. 1.18(b)). Therefore, the computation of $d_{j+1}$ from $d_j$ (equation 1.17) requires at most 4 iterations and the determination of the whole sequence of cracks composing a boundary between two regions is performed in a time proportional to the length of this boundary.

In conclusion, the implicit encoding of a combinatorial pyramid, provides an efficient encoding of the pyramid with a low memory cost and an efficient access to the geometry of segments defining the image's partition. Concerning the encoding of relationships between regions, the combinatorial pyramid framework provides, as the dual graph pyramid framework an encoding of the *meets_each* relationship. Moreover, using the explicit encoding of the orientation provided by combinatorial maps, *contains* and *inside* relationships may be characterized locally [33]. Table 1.2 summarize the relationships between regions encoded by the different models described in this chapter.

| | meets_exists | meets_each | contains/inside | Composed of |
|---|---|---|---|---|
| RAG | ✓ | ✕ | ✕ | ✕ |
| Combinatorial maps | ✓ | ✓ | ✓ | ✕ |
| Simple Graph Pyramids | ✓ | ✕ | ✕ | ✓ |
| Dual Graph Pyramids | ✓ | ✓ | Global calculus? | ✓ |
| Combinatorial Pyramids | ✓ | ✓ | Local calculus | ✓ |

Table 1.2: Relationships available(✓) or not available (✕) using models described in this chapter.

## 1.5 Conclusion

# Bibliography

[1] P. Burt, T.-H. Hong, and A. Rosenfeld, "Segmentation and estimation of image region properties through cooperative hierarchial computation," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 11, no. 12, pp. 802–809, December 1981.

[2] M. Bister, J. Cornelis, and A. Rosenfeld, "A critical view of pyramid segmentation algorithms," *Pattern Recognit Letter.*, vol. 11, no. 9, pp. 605–617, September 1990.

[3] W. G. Kropatsch, "From equivalent weighting functions to equivalent contraction kernels," in *Digital Image Processing and Computer Graphics (DIP-97): Applications in Humanities and Natural Sciences*, E. Wenger and L. I. Dimitrov, Eds., vol. 3346. SPIE, 1998, pp. 310–320.

[4] A. Rosenfeld and C.-Y. Sher, "Detecting image primitives using feature pyramids," *Journal of Information Sciences*, vol. 107, pp. 127–147, 1998.

[5] A. Montanvert, P. Meer, and A. Rosenfeld, "Hierarchical image analysis using irregular tessellations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 307–316, APRIL 1991.

[6] P. Meer, "Stochastic image pyramids," *Computer Vision Graphics Image Processing*, vol. 45, pp. 269–294, 1989.

[7] J. M. Jolion and A. Montanvert, "The adaptative pyramid: A framework for 2d image analysis," *Computer Vision, Graphics, and Image Processing*, vol. 55, no. 3, pp. 339–348, May 1992.

[8] J.-M. Jolion, "Data driven decimation of graphs," in *Proceedings of $3^{rd}$ IAPR-TC15 Workshop on Graph based Representation in Pattern Recognition*, J.-M. Jolion, W. Kropatsch, and M. Vento, Eds., Ischia-Italy, May 2001, pp. 105–114.

[9] S.Thorpe, D.Fize, and C. Marlot, "Speed of precessing in the human visual system," *Nature*, vol. 381, pp. 520–522, 1996.

[10] R. Diestel, *Graph Theory*, ser. Graduate Texts in Mathematics. Springer-Verlag, 1997, vol. 173.

[11] W. G. Kropatsch, Y. Haxhimusa, Z. Pizlo, and G. Langs, "Vision pyramids that do not grow too high," *Pattern Recognition Letters*, vol. 26, no. 3, pp. 319 – 337, 2005, in Memoriam: Azriel Rosenfeld. [Online]. Available: http://www.sciencedirect.com/science/article/B6V15-4F29HX3-2/2/8b603d5889291cbfc062b27cc75facdf

[12] M. Burge and W. Kropatsch, "A minimal line property preserving representation of line images," *Computing*, vol. 62, pp. 355–368, 1999.

[13] R. Englert and W. G. Kropatsch, “Image Structure From Monotonic Dual Graph Contraction,” in *Applications of Graph Transformations with Industrial Relevance*, ser. Lecture Notes in Computer Science, M. Nagl, A. Schürr, and M. Münch, Eds., vol. Vol. 1799. Kerkrade, Netherlands: Springer, Berlin Heidelberg, New York, 2000, pp. 297–308.

[14] L. Brun and W. Kropatsch, “Introduction to combinatorial pyramids,” in *Digital and Image Geometry*, ser. LNCS, R. K. G. Bertrand, A. Imiya, Ed. Springer Verlag, 2001, vol. 2243, pp. 108–127.

[15] V. Kovalevsky, “Finite topology as applied to image analysis,” *Computer Vision, Graphics, and Image Processing*, vol. 46, pp. 141–161, 1989.

[16] L. Brun, M. Mokhtari, and J. P. Domenger, “Incremental modifications on segmented image defined by discrete maps,” *Journal of Visual Communication and Image Representation*, vol. 14, pp. 251–290, 2003.

[17] D. Randell, C. Z, and A. Cohn, “A spacial logic based on regions and connections,” in *Principle of Knowledge Representation and Reasoning: Proceedings 3rd International Conference*, B. Nebel, W. Swartout, and C. Rich, Eds., Cambridge MA, 1992, pp. 165–176.

[18] K. Shearer, H. Bunke, and S. Venkatesh, “Video indexing and similarity retrieval by largest common subgraph detection using decision trees,” *Pattern Recognition*, vol. 34, pp. 1075–1091, 2001.

[19] L. Brun and M. Mokhtari, “Graph based representations in different application domains,” in $3^{rd}$ *IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, J. M. Jolion, W. Kropatsch, and M. Vento, Eds., IAPR-TC15. Ischia Italy: CUEN, May 2001, pp. 115–124, invited conference.

[20] D. S. Tweed and A. D. Calway, “Integrated segmentation next term and depth ordering of motion layers in image sequences,” *Image and Vision Computing*, vol. 20, no. 9, pp. 709–723, August 2003.

[21] D. Willersinn and W. G. Kropatsch, “Dual graph contraction for irregular pyramids,” in *International Conference on Pattern Recogntion D: Parallel Computing*. Jerusalem, Israel: International Association for Pattern Recognition, 1994, pp. 251–256.

[22] P. Lienhardt, “Topological models for boundary representations: a comparison with $n$-dimensional generalized maps,” *Computer-Aided Design*, vol. 23, no. 1, pp. 59–82, 1991.

[23] Y. Bertrand, G. Damiand, and C. Fiorio, “Topological map: Minimal encoding of 3d segmented images,” in $3^{rd}$ *Workshop on Graph-based Representations in Pattern Recognition*, J. M. Jolion, W. Kropatsch, and M. Vento, Eds., IAPR-TC15. Ischia(Italy): CUEN, May 2001, pp. 64–73.

[24] L. Brun and W. Kropatsch, “The construction of pyramids with combinatorial maps,” Institute of Computer Aided Design, Vienna University of Technology, lstr. 3/1832,A-1040 Vienna AUSTRIA, Tech. Rep. 63, June 2000. [Online]. Available: http://www.prip.tuwien.ac.at/

[25] ——, “Contraction kernels and combinatorial maps,” in $3^{rd}$ *IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, J. M. Jolion, W. Kropatsch, and M. Vento, Eds., IAPR-TC15. Ischia Italy: CUEN, May 2001, pp. 12–21.

[26] C. Fiorio, “Approche interpixel en analyse d’images : une topologie et des algorithmes de segmentation,” Thèse de doctorat, Université Montpellier II, 24 novembre 1995.

[27] J. P. Braquelaire and L. Brun, “Image segmentation with topological maps and inter-pixel representation,” *Journal of Visual Communication and Image representation*, vol. 9, no. 1, pp. 62–79, 1998.

[28] W. G. Kropatsch, “Equivalent Contraction Kernels and The Domain of Dual Irregular Pyramids,” Institute f. Automation 183/2, Dept. for Pattern Recognition and Image Processing, TU Wien, Austria, Tech. Rep. PRIP-TR-42, 1995. [Online]. Available: www.prip.tuwien.ac.at/ftp/pub/publications/trs/tr42.ps.gz

[29] ——, “Building Irregular Pyramids by Dual Graph Contraction,” *IEE-Proc. Vision, Image and Signal Processing*, vol. Vol. 142, no. No. 6, pp. pp. 366–374, December 1995.

[30] L. Brun and W. Kropatsch, “Combinatorial pyramids,” in *IEEE International conference on Image Processing (ICIP)*, Suvisoft, Ed., vol. II. Barcelona: IEEE, September 2003, pp. 33–37.

[31] L. Brun, M. Mokhtari, and F. Meyer, “Hierarchical watersheds within the combinatorial pyramid framework,” in *Proc. of DGCI 2005*, IAPR-TC18. LNCS, 2005, pp. 34–44.

[32] L. Brun, “Traitement d’images couleur et pyramides combinatoires,” Habilitation à diriger des recherches, Université de Reims, 2002.

[33] L. Brun and W. Kropatsch, “Contains and inside relationships within combinatorial pyramids,” *Pattern Recognition*, vol. 39, no. 4, pp. 515–526, April 2006.