Receptive fields within the Combinatorial Pyramid framework

Luc Brun [a,*], Walter Kropatsch [b,1]

[a] *Laboratoire d'Études et de Recherche en Informatique(EA 2618)*
Université de Reims - France

[b] *Institute for Computer-aided Automation*
Pattern Recognition and Image Processing Group
Vienna Univ. of Technology- Austria

Abstract

A hierarchical structure is a stack of successively reduced image representations. Each basic element of a hierarchical structure is the father of a set of elements in the level below. The transitive closure of this father-child relationship associates to each element of the hierarchy a set of basic elements in the base level image representation. Such a set, called a receptive field, defines the embedding of one element on the original image. Combinatorial pyramids are defined as a stack of successively reduced combinatorial maps, each combinatorial map being defined by two permutations acting on a set of half edges named darts. The basic element of a combinatorial pyramid is thus the dart. This paper defines the receptive field of each dart within a combinatorial pyramid and study the main properties of these sets.

Key words: Graphs, Combinatorial maps, Irregular Pyramids, Receptive fields

1 Introduction

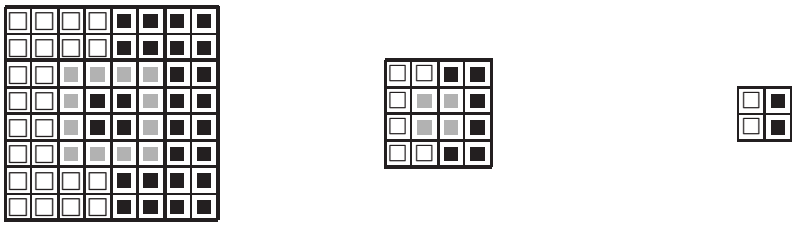
Regular image pyramids have been introduced 1981/82 [1] as a stack of images with exponentially reduced resolution. Each image of this sequence is called

* Corresponding author
Email addresses: luc.brun@univ-reims.fr (Luc Brun), krw@prip.tuwien.ac.at (Walter Kropatsch).

[1] This Work was supported by the Austrian Science Foundation under grants P14445-MAT and P14662-INF.

a *level*. Such Pyramids present several interesting properties within the image processing and analysis framework such as [2]: The reduction of noise, the processing of local and global features within the same frame and the efficiency of many computations on this structure. Using the neighborhood relationships defined on each image the *Reduction window* relates each pixel of the pyramid with a set of pixels defined in the level below. The pixels belonging to one reduction window are the *children* of the pixel which defines it. The value of each father is computed from the one of its children using a *Reduction function*. A regular pyramid is thus defined by the ratio $N \times N/q$ where $N \times N$ is the size of the reduction window, and q the ratio between the size of two consecutive images in the pyramid.

The father-child relationship defined by the reduction window maybe extended by transitivity down to the base level image. **The set of children of one pixel in the base level** is named its *receptive field* (RF) and defines the embedding of this pixel on the original image. Using the father-child relationship global properties of a receptive field $RF(v)$ with a diameter m may be computed in $\mathcal{O}(\log(m))$ parallel processing steps thanks to local calculus. However, receptive fields defined within the regular pyramid framework are not necessarily connected [2]. Furthermore, the adjacency of two pixels v and w defined at level k may not be easily interpreted on the base level image. This last drawback is illustrated in Fig. 1 where the initial 8×8 image is reduced by a $2 \times 2/4$ regular pyramid using the mean gray level as reduction function. Each black pixel in the central region belongs to a 2×2 reduction window with 3 gray pixels. These pixels are thus mapped onto a gray father and the black region which was disconnected at level 0 becomes connected at level 1. Finally, the boundary between the receptive fields $RF(v)$ and $RF(w)$ associated to this adjacency at level k may be disconnected and even incomplete (see [2] for more details).

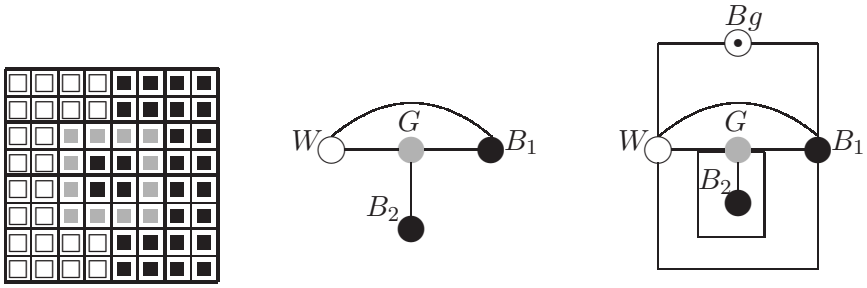


(a) Initial Image (b) level 1 (c) level 2

Fig. 1. A $2 \times 2/4$ regular pyramid. The central black region is removed from level 0 to 1 due to the fixed decimation ratio and the reduce size of reduction windows.

Irregular pyramids, first introduced by Meer, Montanvert and Jolion [3] are defined as a stack of successively reduced simple graphs (i.e. graphs without double edges nor self-loops). The base level graph may be built from a

sampling grid using one pixel adjacency such as the 4−neighborhood. Each graph of the hierarchy is built from the graph below by selecting a set of vertices named surviving vertices and mapping each non surviving vertex to a surviving one [3]. This mapping induces a father-child relationship between a surviving vertex and the set of non surviving vertices mapped to it. The reduction window of one surviving vertex is then defined as its set of children. The receptive field of one surviving vertex is defined by the transitive closure of the father-child relationship. Using this reduction scheme, the receptive field of each vertex in the hierarchy is a **connected set of vertices in the base level graph**. However, using simple graphs, the adjacency between two vertices is encoded by only one edge while the receptive fields of two vertices may share several boundary segments. An edge in the hierarchy may thus encode a non-connected set of boundaries between the associated receptive fields. Moreover, the lack of self-loops in simple graphs does not allow to differentiate an adjacency relationship between two receptive fields from an inclusion relationship. These two drawbacks are illustrated in Fig. 2(b) which represents the top of a simple graph pyramid encoding the connected components of Fig. 2(a). The two boundaries between the white region (W) and the black one on the right of the image (B_1) are encoded by only one edge. Moreover, the adjacencies between the gray region (G) and the two black ones (B_1 and B_2) are encoded in the same way by a simple edge. Therefore, these two different types of adjacency can not be distinguished at the top of a simple graph pyramid.



(a) Initial image

(b) Simple graph pyramid

(c) Dual graph or combinatorial pyramids

Fig. 2. Encoding of the connected components of a 8×8 image (a) by a simple graph pyramid (b) and the Dual Graph or Combinatorial Pyramids (c). The vertex B_g in (c) encodes the background of the image.

The last two drawbacks may be overcome by using the Dual graph pyramids introduced by Kropatsch and Willersinn [4]. Using Kropatsch's reduction scheme, the reduction operation is encoded by edge contractions [4]. This operation contracts one edge and its two end points into a single vertex. It corresponds to the edge collapse operation used by Hoppe et al. [5] to simplify

triangular meshes. The contraction of a graph reduces the number of vertices while maintaining the connections to other vertices. As a consequence some redundant edges such as self-loops or double edges may occur, some encode relevant topological relations (e.g. an island in a lake) others can be removed without any harm to the involved topology. These redundant edges may be characterized in the dual of the contracted graph. The removal of such edges is called a dual decimation step. Since the reduction scheme requires both features of the initial graph and its dual, such pyramids are called *Dual graph pyramids*. Within such hierarchies, each receptive field is a **connected set of vertices in the base level**. Moreover, each edge between two vertices encodes an unique connected boundary between the associated receptive fields. Finally, the use of self-loops within the hierarchy allows to differentiate adjacency relationships between receptive fields from inclusion relations. These two properties are illustrated in Fig. 2(c) which represents the top of a dual graph pyramid encoding the connected components of Fig. 2(a). The inclusion of the central black region within the gray one is encoded by the self-loop which surrounds the vertex B_2 associated to this region. Moreover, the two boundaries between the white region (W) and the black one on the right of the image (B_1) are encoded by two edges, each edge being associated to one boundary.

The scale-space [6] and discrete wavelet [7] transforms are another stream of hierarchical image expression. Within this stream, the hierarchical description of the image is obtained by the successive application of a sequence of filters ψ_j. Each image at level j is defined as a sub-sampling of the image at level $j-1$. Note that from this point of view the regular pyramid may be understood as a particular case of scale-space representation. Indeed, within the regular pyramid framework each image is defined as a sub-sampling of the image below and the value of each pixel is usually computed by applying a given filter on its reduction window. Given a particular vision task, applications using scale-space or discrete wavelet transforms define a particular filter according to the task to perform. Image processing algorithms are then applied on this multi-scale description of the image. Applications to image compression, denoising, edge detection, optical flow and stereo-vision show the great potential of this stream [7]. The main advantages of Pyramid transforms beside scale-space representation are two fold: First, within the pyramid transform the decimation process may be adapted during the construction of the pyramid according to the content of the image. For example, within the segmentation framework one can first reduce the initial image according to low level criteria in order to produce a partition into regions fulfilling a given homogeneity criteria. Higher level reduction process may then be used from this intermediate partition to group regions into objects. Within the discrete wavelet or scale-space transform framework, the initial filters are chosen according to the application but are not usually modified during the construction of the hierarchical representation. Secondly, the pyramidal framework allows to encode the topological

relationships between the regions of the hierarchy. Such an encoding is not directly provided by scale-space or discrete wavelets transforms.

The remaining of this paper is organized as follows: The combinatorial map model is presented in Section 2 together with the expected advantages of this model within the Pyramid framework. We present in Section 3 the construction scheme of a combinatorial pyramid. The notion of dart's and vertex's reduction window are presented in Section 4. Section 5 defines the notion of dart's receptive field within the combinatorial pyramid framework and states its major properties used for folding and unfolding of the whole pyramid (Section 6).

2 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 [8]. Recent trends in combinatorial maps apply this framework to the segmentation of 3D images [9] and the encoding of hierarchies [10,11].

The remaining of this paper will be based on 2D combinatorial maps 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 3(a) demonstrates the derivation of a combinatorial map from a plane graph $G = (V, E)$ corresponding to a 3×3 pixel grid. 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 α. A second permutation σ encodes the set of darts encountered when turning counterclockwise 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 σ, α are two permutations defined on $\mathcal{D}$ such that α is an involution:

$$\forall d \in \mathcal{D} \quad \alpha^2(d) = d \tag{1}$$

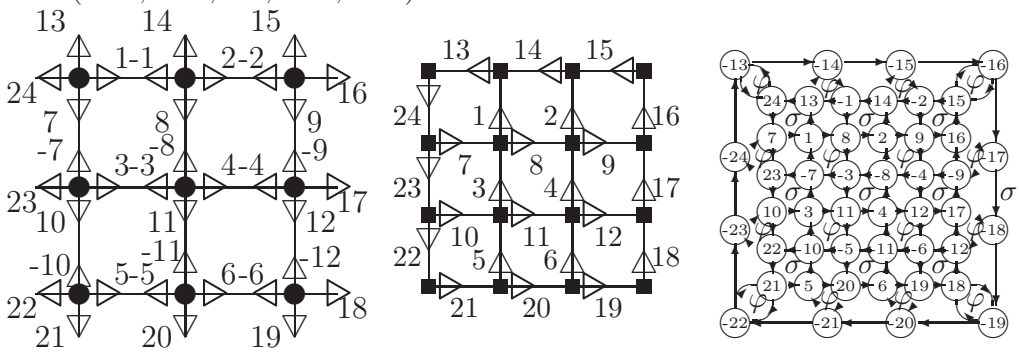
If the darts are encoded by positive and negative integers, the involution α may be implicitly encoded by the sign (Figure 3(a)).

Given a dart d and a permutation π, the π-orbit of d denoted by $\pi^*(d)$ is the

series of darts $(\pi^i(d))_{i\in\mathbb{N}}$ defined by the successive applications of π on the dart d. The σ and α orbits of a dart d will be respectively denoted by $\sigma^*(d)$ and $\alpha^*(d)$.

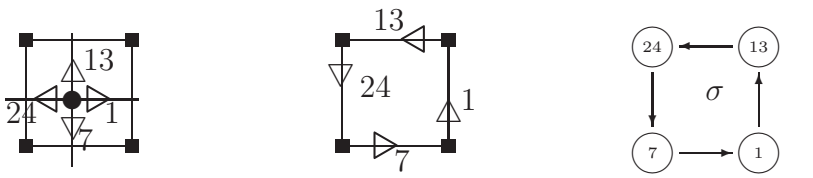
Fig. 3 illustrates the encoding $G = (\{\bullet\}, \{\bullet - \bullet\})$ of a 3×3 4-connected discrete grid by a combinatorial map. Each vertex of the initial combinatorial map (Fig. 3(a)) encodes a pixel of the grid. The σ-orbit of one vertex encodes its adjacency relationships with neighboring vertices. Let us consider the dart 1 in Fig. 3(a) (see also Fig. 4(a)). Turning counterclockwise around the central vertex we encounter the darts $1, 13, 24$ and 7. We have thus $\sigma(1) = 13$, $\sigma(13) = 24$, $\sigma(24) = 7$ and $\sigma(7) = 1$. The σ-orbit of 1 is thus defined as $\sigma^*(1) = (1, 13, 24, 7)$.

The permutation α being implicitly encoded by the sign in Fig. 3, we have $\alpha^*(1) = (1, -1)$. The α successors of the darts 13 to 24 are not shown in Fig. 3(a) in order to not overload it. These darts encode the adjacency relationships between the border pixels of the grid and its background. The σ orbit of the background vertex is equal to the sequence of darts from -13 to -24: $(-13, -14, \ldots, -23, -24)$.



(a) G (b) $\overline{G}$ (c) $\mathcal{OG}$

Fig. 3. A 3×3 grid encoded by a combinatorial map



(a) $\sigma^*(1)$ in G (b) $\sigma^*(1)$ in $\overline{G}$ (c) $\sigma^*(1)$ in $\mathcal{OG}$

Fig. 4. The σ-orbit $\sigma^*(1)$ encoding the top-left pixel represented in three equivalent representation : The initial combinatorial map G (a), its dual $\overline{G}$ (b) and the oriented graph $\mathcal{OG}$ (c).

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 orbits of the permutation φ encode the set of darts encountered when turning around a face of G. Note that, using a counter-clockwise

orientation for permutation σ, each dart of a φ-orbit has its associated face on its right (see e.g. the φ-orbit $\varphi^*(1) = (1, 8, -3, -7)$ in Figure 3(a)).

The dual combinatorial map $\overline{G} = (\{\blacksquare\}, \{\blacksquare - \blacksquare\})$ is shown in Fig. 3(b). We also do not show the α-successor of the positive darts on this figure to not overload it. Each vertex of this dual map may be associated to a corner of a pixel. The top-left corner of the pixel represented in Fig. 4(b) is for example encoded by the φ orbit : $\varphi^*(24) = (24, -13)$. Moreover, each dart may be understood in this combinatorial map as an oriented crack, i.e. as a side of a pixel with an orientation. For example, the dart 1 in Fig. 3(b) encodes the right side of the upper-left pixel oriented from bottom to top (Fig. 4(b)). The φ, α and σ orbits of a dart may thus be respectively understood as elements of dimensions 0, 1 and 2. A combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$ encoding a planar sampling grid may thus be associated to the cellular complex:

$$C = (E_0 \cup E_1 \cup E_2, B, dim) \tag{2}$$

with $E_0 = \{\varphi^*(d),\ d \in \mathcal{D}\}, E_1 = \{\alpha^*(d),\ d \in \mathcal{D}\}$, $E_2 = \{\sigma^*(d),\ d \in \mathcal{D}\}$, $B = (E_0 \times E_1) \cup (E_1 \times E_2) \cup (E_0 \times E_2)$ and $dim(b) = i$ for all $b \in E_i$, $i \in \{0, 1, 2\}$.

Using Fig. 4, the $2D$ cell encoding the top left pixel is encoded by the σ-orbit $\sigma^*(1) = (1, 13, 24, 7) \in E_2$ (Fig. 4(a)). This $2D$ cell is bounded by the $1D$ cells defined by $\{\alpha^*(1), \alpha^*(13), \alpha^*(24), \alpha^*(7)\} \subset E_1$ (Fig. 4(b)). Finally, each $1D$ cell $\alpha^*(d)$ is bounded by the two $0D$ cells $\varphi^*(d)$ and $\varphi^*(\alpha(d))$. The $1D$ cell $\alpha^*(24)$ is for example, bounded by $\varphi^*(24) = (24, 13) \in E_0$ and $\varphi^*(-24) = (-24, 7, 23) \in E_0$ (Fig. 4(b) and 3(b)). A discrete topology [12] based on cellular complexes may thus be associated to combinatorial maps (see [8] for a more precise study).

Fig. 3(c) illustrates the $\sigma - \varphi$ representation of a combinatorial map. Within this alternative representation, a combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$ is described by an oriented planar graph $\mathcal{OG} = (V, E)$. The set V of vertices of $\mathcal{OG}$ is equal to the set of darts $\mathcal{D}$ and an oriented edge $e \in E$ connects two vertices d_1 and d_2 iff either $d_2 = \sigma(d_1)$ or $d_2 = \varphi(d_1)$ (see e.g. the encoding of the top-left vertex in Figure 4(c)). Using this representation, the σ and φ orbits of the combinatorial map are represented by the faces of the oriented graph $\mathcal{OG}$. Note that each vertex of $\mathcal{OG}$ has two incoming arcs (its σ and φ predecessors) and two outgoing ones (its σ and φ successors).

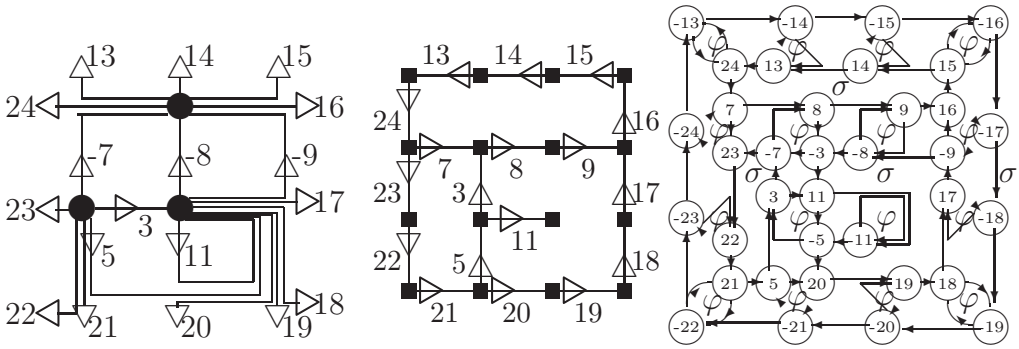
As mentioned above the dual combinatorial map may be simply computed by composing the permutations σ and α. This dual map may thus be implicitly encoded. This implicit encoding allows to reduce both the memory requirements and the execution times since only one data structure needs to be stored and processed. Moreover the combinatorial map formalism is formally defined in any dimensions. Algorithms defined within the $2D$ combinatorial pyramid framework may thus be extended to higher dimensions(see e.g. [8]). This hypothesis is confirmed by some recent results from Damiand and Lienhardt [13]. Finally, the combinatorial map formalism encodes explicitly the orientation of edges around each vertex. This additional feature allows to encode fine relationships on the partition. For example, an encoding by a combinatorial map of the graph represented in Fig. 2(c) would encode the fact that turning counterclockwise around the region encoded by the vertex B_1 we encounter the sequence of regions W, G, W and Bg. This feature is not explicitly encoded within the dual graph framework.

3 Combinatorial Pyramids

The aim of combinatorial pyramids is to combine the advantages of combinatorial maps (Section 2.1) with the reduction scheme defined by Kropatsch [14] (see also Section 1). A combinatorial pyramid is thus defined by an initial combinatorial map successively reduced by a sequence of contraction or removal operations. In order to preserve the connectivity among the elements chosen to survive to the higher level we forbid the removal of bridges and the contraction of self-loops. A self-loop in the initial combinatorial map corresponds to a bridge in its dual and vice-versa [15]. In the same way, a contraction operation in the initial combinatorial map is equivalent to a removal operation performed in its dual. Therefore, the exclusion of bridges and self-loops from respectively removal and contraction operations corresponds to a same constraint applied alternatively on the dual combinatorial map and the original one.

In order to avoid the contraction of self-loops, the set of edges to be contracted must form a forest of the initial combinatorial map. The graph of a combinatorial map is a forest if it does not contain a cycle. A more formal definition may be found in [10]. A set of edges to be contracted satisfying the above requirement is called a contraction kernel:

Definition 1 Contraction Kernel



(a) $G_1 = G/K_1$ (b) $\overline{G_1} = \overline{G/K_1}$ (c) $\mathcal{OG}_1$

Fig. 5. Reduction of the initial grid displayed in Fig. 3 by the contraction kernel K_1

Given a connected combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$ the set $K \subset \mathcal{D}$ will be called a contraction kernel iff K is a forest of G.

The set $\mathcal{SD} = \mathcal{D} - K$ is called the set of surviving darts.

Given a contraction kernel K, we denote by $\mathcal{CC}(K)$ its set of connected components. Since K is a forest, each $\mathcal{T} \in \mathcal{CC}(K)$ is a tree. Intuitively, a tree $\mathcal{T} \in \mathcal{CC}(K)$ collapses into one vertex, a connected set of vertices of the initial combinatorial map.

The contraction of a combinatorial map by a contraction kernel may induce the creation of redundant edges. These edges named double edges and direct-self-loops are respectively characterized by $\varphi^2(d) = d$ and $\sigma(d) = \alpha(d)$ where d is one of the darts of the edge $\alpha^*(d)$. Both double edges and direct self-loops may be removed by using a removal kernel defined as a forest of the dual combinatorial map. This last constraint insures that no self-loop will be contracted in the dual combinatorial map and thus that no bridge may be removed in the initial one. The contracted combinatorial map may thus be simplified using parallel edge removals.

3.1 Example of contraction and simplification

Fig. 5 illustrates a contraction of the initial combinatorial map represented in Fig. 3 by a contraction kernel K_1 defined by the trees $\alpha^*(1,2), \alpha^*(4,12,6)$ and $\alpha^*(10)$. Since each initial vertex is incident to a contracted edge this forest spans the initial combinatorial map and we obtain 3 surviving vertices encoding 3 regions. The contraction of the first row of the image by the tree $\alpha^*(1,2)$ is represented in Fig. 6. The same row, is reprensented in the dual combinatorial map $\overline{G}$ in Fig. 7.

Double edges encode a same adjacency relationship between two vertices. For example, the edges $\alpha^*(8)$ and $\alpha^*(9)$ in Fig. 5(a) encode both an adjacency

(a) Ininitial row (b) Contrated one

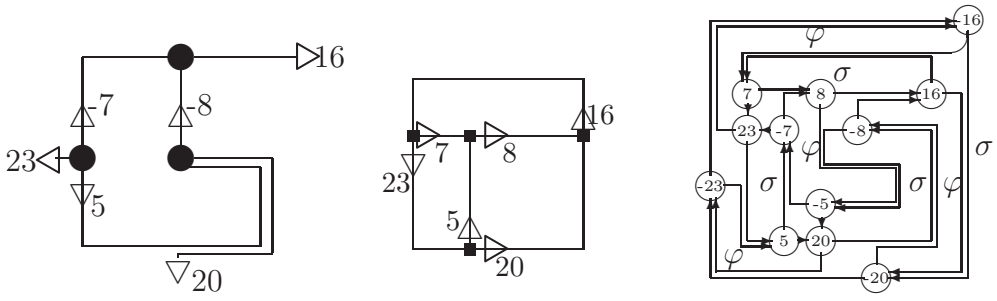
Fig. 6. Contraction of the first row represented in Fig. 3(a). The contracted edges are represented by thick lines.

(a) Initial row (b) Contracted one

Fig. 7. Contraction of the first row represented in the dual combinatorial map (Fig. 3(b)). The edges to be removed are represented by thick lines.

relationship between the top vertex and the center one. Such edges correspond to an artificial split of a boundary between two regions (see Fig. 5(b)). On the other hand, direct self-loops, such as $\alpha^*(11)$ (Fig. 5(a)) may be interpreted in the dual combinatorial map as inner-boundaries of a face (Fig. 5(b)).

Fig. 8 shows the combinatorial map deduced from Fig. 5 and simplified by the removal kernel $K_2 = \{\alpha^*(15, 14, 13, 24), \alpha^*(9), \alpha^*(11, 3), \alpha^*(19, 18, 17), \alpha^*(22, 21)\}$. Note that given a sequence of double edges, the choice of the surviving edge is arbitrary. For example, a choice of the tree $\alpha^*(20, 19, 18)$ instead of $\alpha^*(19, 18, 17)$ would lead to an equivalent simplified combinatorial map with a surviving edge equal to $\alpha^*(17)$ instead of $\alpha^*(20)$.



(a) $(G/K_1) \setminus K_2$ (b) $\overline{(G/K_1) \setminus K_2}$ (c) $\mathcal{OG}_2$

Fig. 8. Reduction of the contracted combinatorial map displayed in Fig. 5 by the removal kernel K_2

3.2 Connecting walks

Contraction and removal kernels specify the set of edges which must be contracted or removed. The creation of the reduced combinatorial map from a contraction or a removal kernel is performed in parallel by using connecting walks [11]. Given a combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$, a kernel K and a surviving dart $d \in \mathcal{SD} = \mathcal{D} - K$, the connecting walk associated to d is equal to:

$$CW(d) = d, \pi(d), \ldots, \pi^{n-1}(d) \text{ with } n = Min\{p \in \mathbb{N}^* \mid \pi^p(d) \in \mathcal{SD}\} \quad (3)$$

where π is associated with φ if K is a contraction kernel and with σ otherwise.

Given a kernel K and a surviving dart $d \in \mathcal{SD}$, such that $CW(d) = d.d_1 \ldots d_p$, the successor of d within the reduced combinatorial map $G' = (\mathcal{SD}, \sigma', \alpha)$ is retrieved from $CW(d)$ by the following equations [11]:

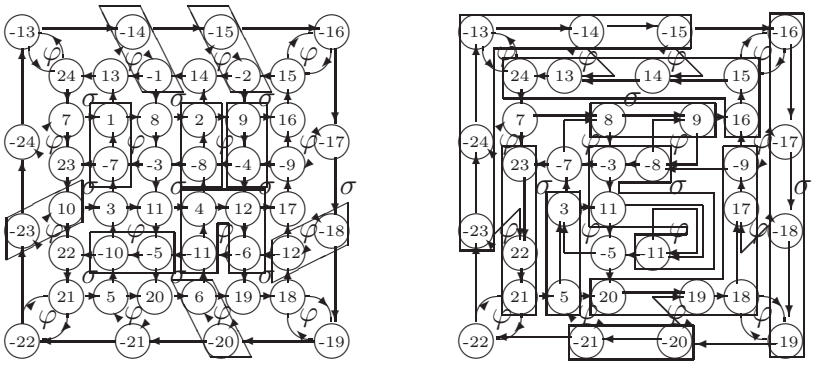
$$\begin{aligned} \varphi'(d) &= \varphi(d_p) \text{ if } K \text{ is a contraction kernel} \\ \sigma'(d) &= \sigma(d_p) \text{ if } K \text{ is a removal kernel} \end{aligned} \quad (4)$$

Note that, if K is a contraction kernel, the connecting walk $CW(d)$ allows to compute $\varphi'(d)$. The σ-successor of d within the contracted combinatorial map may be retrieved from $CW(\alpha(d)) = \alpha(d).d'_1, \ldots, d'_p$. Indeed, we obtain by using Eq. (1) and Eq. (4): $\varphi'(\alpha(d)) = \sigma'(\alpha(\alpha(d))) = \sigma'(d) = \varphi(d'_p)$. We may alternatively consider the sequence $d.CW^*(\alpha(d))$ where $CW^*(\alpha(d))$ denotes the sequence $CW(\alpha(d))$ without its first dart $\alpha(d)$. In this case, using Eq. (4) the σ successor of a surviving dart d is provided by the last dart of $CW(d)$ if K is a removal kernel and by the last dart of $d.CW^*(\alpha(d))$ if K is a contraction kernel.

3.3 Illustration of Connecting walks

Fig. 9 shows the connecting walks defined by K_1 and K_2 superimposed to the oriented graphs $\mathcal{OG}$ (Fig. 3(c)) and $\mathcal{OG}_1$ (Fig. 5(c)) respectively associated to $G = (\mathcal{D}, \sigma, \alpha)$ and $G_1 = G/K_1 = (\mathcal{SD}_1, \sigma_1, \alpha)$. Let us consider the surviving dart -5 of G_1 (Fig. 9(a)). Since K_1 is a contraction kernel $CW_1(-5)$ is equal to the sequence of non-surviving φ successors of -5. Since $\varphi(-5) = -10 \in K_1$ and $\varphi(-10) = 3 \in \mathcal{SD}_1$, we have $CW_1(-5) = -5. -10$ (Fig. 9). Given $CW_1(-5)$, we obtain by Eq. (4): $\varphi_1(-5) = \sigma_1(5) = \varphi(-10) = 3$ (Fig. 5).

In the same way, let us now consider the combinatorial map $G_2 = G_1 \setminus K_2 =$



(a) connecting walks in $\mathcal{OG}$ defined by K_1

(b) Connecting walks in $\mathcal{OG}_1$ defined by K_2

Fig. 9. Connecting walks defined by K_1 (a) and K_2(b).

$(\mathcal{SD}_2, \sigma_2, \alpha)$ and the surviving dart $5 \in \mathcal{SD}_2$ (Fig. 9(b)). Since K_2 is a removal kernel, the connecting walk of 5 is defined as the sequence of non-surviving σ-successors of 5. Since $\sigma_1(5) = 3 \in K_2$ and $\sigma_1(3) = -7 \in \mathcal{SD}_2$ (Fig. 5) we have: $CW_2(5) = 5.3$ (Fig. 9(b)). Moreover, using $CW_2(5)$ and Eq. (4) we have $\sigma_2(5) = \sigma_1(3) = -7$ (Fig. 8).

4 Reduction Window of darts and of vertices

Given an initial combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$ and a kernel K, we designed [11] two algorithms to traverse connecting walks. These algorithms may be used in association with Eq. (4) to compute the reduced combinatorial map $G' = (\mathcal{SD} = \mathcal{D} - K, \sigma', \alpha)$. However, connecting walks may also be used to relate one combinatorial map of the pyramid with the one below. Indeed, using Eq. (4) each connecting walk associates one surviving dart d to either $\sigma'(d)$ or $\varphi'(d)$ according to the type of the kernel K. Since the reduced combinatorial map is defined by the permutation σ', we define the reduction window of a surviving dart d as the sequence of non surviving darts which connects d to $\sigma'(d)$ in G. This sequence is thus defined by (Section 3):

$$RW(d) = \begin{cases} CW(d) & \text{if } K \text{ is a removal kernel} \\ dCW^*(\alpha(d)) & \text{if } K \text{ is a contraction kernel.} \end{cases} \quad (5)$$

A dart's reduction window may thus be computed using the algorithms which traverse connecting walks [11].

However, within the Irregular Pyramid framework, the reduction window relates one surviving vertex with a set of non-surviving vertices in the graph

below (Section 1). This vertex set is connected and forms a tree structure. Within the Combinatorial Pyramid framework, a surviving vertex is defined by its σ'-orbit. Let us consider such an orbit $\sigma'^*(d_1) = (d_1, \ldots, d_p)$. The **reduction window of the vertex** $\sigma'^*(d_1)$ denoted by $R_{\sigma'^*(d_1)}$ is defined as the concatenation of the reduction windows of each of its darts:

$$R_{\sigma'^*(d_1)} = RW(d_1).....RW(d_p) \tag{6}$$

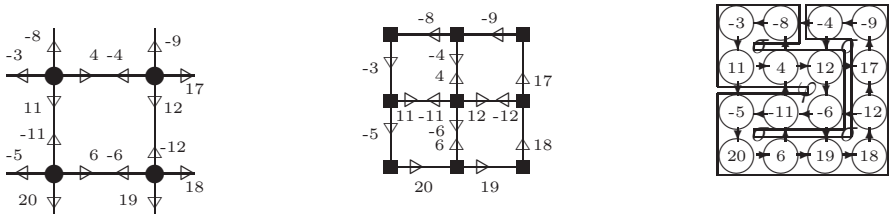
Note that $R_{\sigma'^*(d_1)}$ is defined, as $\sigma'^*(d_1)$, up to a circular permutation on $(d_1, \ldots, d_p)$.

Within the Combinatorial Pyramid framework a vertex's reduction window is thus defined as a sequence of darts. We have shown (see proof in [16]) that the vertex's reduction window $R_{\sigma'^*(d_1)}$ contains all the darts of each non surviving vertex mapped to $\sigma'^*(d_1)$ and define a connected set of vertices. It additionally defines a cycle in the graph $\mathcal{OG}$ (Fig. 10(c)).

Our definition of a vertex's reduction window corresponds thus to the usual notion of reduction window such as the one defined by Willersinn and Kropatsch [4]. The edge associated to each dart of $R_{\sigma'^*(d_1)}$ may be classified into two categories: We say that **an edge** $\alpha^*(d)$ **is an inner boundary of the reduction window** $R_{\sigma'^*(d_1)}$ if it connects two vertices included in $R_{\sigma'^*(d_1)}$. Conversely an edge is said to be an **external boundary** of $R_{\sigma'^*(d_1)}$ if it connects one vertex in $R_{\sigma'^*(d_1)}$ with another vertex of G not included in $R_{\sigma'^*(d_1)}$.

4.1 Illustration of a vertex's reduction window

Fig. 10 illustrates three alternative representations of the reduction window associated to the contracted vertex $\sigma'^*(-8)$ (see central vertex in Fig. 5a). In the reduced graph G_1 (Fig. 5) the vertex $\sigma'^*(-8)$ is defined by the sequence of darts -8.-3.11.-11.-5.20.19.18.17.-9 . Since K_1 is a contraction kernel the reduction window of each dart $d \in R_{\sigma'^*(-8)}$ is equal to $d.CW^*(\alpha(d))$ (Eq. (5)).



(a) $R_{\sigma'^*(-8)}$ in G (b) $R_{\sigma'^*(-8)}$ in $\overline{G}$ (c) $R_{\sigma'*(-8)}$ in $\mathcal{OG}$

Fig. 10. The reduction window $R_{\sigma'^*(-8)}$ defined by contraction kernel $K_1 \subset \mathcal{D}$

Using Eq. (6), the reduction window $R_{\sigma'^*(-8)}$ is defined by:

$$R_{\sigma'^*(-8)} = \underline{-8}.\underline{-3}.\underline{11}.\underline{4}.\underline{12}.\underline{-6}.\underline{-11}.\underline{-5}.\underline{20}.\underline{6}.\underline{19}.\underline{18}.\underline{-12}.\underline{17}.\underline{-9}.\underline{-4} \qquad (7)$$

where each sequence $d.CW^*(\alpha(d))$ with $d \in \sigma'^*(-8)$ is underlined (cf. Fig. 9(a)).

The reduction window $R_{\sigma'^*(-8)}$ is composed of 4 vertices with 4 edges encoding inner boundaries and 8 edges encoding the external boundary of the reduction window (Section 4). We can note on Eq. (7) that all the edges defining inner boundaries are included in $R_{\sigma'^*(-8)}$. We have indeed, $\alpha^*(4, 11, 6, 12) \subset R_{\sigma'^*(-8)}$.

This last property is a consequence of the fact that a reduction window $R_{\sigma'^*(d_1)}$ contains all the darts of the non-surviving vertices mapped to $\sigma'^*(d_1)$. Indeed, if an edge $\alpha^*(d)$ defines an inner boundary of a vertex's reduction window R both $\sigma^*(d)$ and $\sigma^*(\alpha(d))$ must be included in R. Therefore, d and $\alpha(d)$ must belong to R. Conversely, all darts defining the boundary of a region R cannot have their α-successor in R (see e.g. the darts 17 to 20 in Eq. (7) and Fig. 10).

5 Receptive fields

The notion of connecting walks introduced in Section 3 allows us to build one reduced combinatorial map from an initial one and a contraction or removal kernel. Therefore, given a sequence of kernels $K_1, \ldots, K_n$ and an initial combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$ defined from a planar sampling grid (e.g. the 4−neighborhood) one can define the sequence of reduced combinatorial maps $G_0, G_1, \ldots, G_n$ where $G_i = G_{i-1}/K_i = (\mathcal{SD}_i, \sigma_i, \alpha)$ for each $i \in \{1, \ldots, n\}$. Note that according to the definition of surviving darts (Definition 1) we have $\mathcal{SD}_i = \mathcal{SD}_{i-1} - K_i = \mathcal{D} - \cup_{j=1}^{i} K_j$.

5.1 Connecting dart sequences

Intuitively, one connecting walk $CW_i(d)$ defines the set of darts that we have to traverse in G_{i-1} in order to connect d to $\varphi_i(d)$ if K_i is a contraction kernel, or to connect d to $\sigma_i(d)$ if K_i is a removal kernel (Eq. (4)). Let us consider the sequence of darts $CDS_i(d)$ that we have to traverse in the base level combinatorial map G_0 to connect d to $\varphi_i(d)$ if K_i is a contraction kernel and d to $\sigma_i(d)$ if K_i is a removal kernel. Such a sequence of darts is called a connecting dart sequence(CDS). Moreover, using Definition 2, we have shown[10] that the first dart of $CDS_i(d)$ is d. A connecting dart sequence $CDS_i(d)$ without its first dart will be denoted $CDS_i^*(d)$.

Definition 2 *Given a combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$ and a sequence of contraction or removal kernels $K_1, K_2 \ldots, K_n$. The connecting dart sequences are defined by $CDS_0(d) = d$ for any $d \in \mathcal{D}$ and the following recursive construction:*

For each level i in $\{1, \ldots, n\}$ and for each dart d in $\mathcal{SD}_i$

- *If K_i and K_{i-1} have the same type:*

$$CDS_i(d) = CDS_{i-1}(d_1) \cdots CDS_{i-1}(d_p)$$

- *If K_i and K_{i-1} have different types:*

$$CDS_i(d) = d_1 \cdot CDS^*_{i-1}(\alpha(d_1)) \cdots d_p \cdot CDS^*_{i-1}(\alpha(d_p))$$

Where $(d_1 \ldots d_p)$ is equal to $CW_i(d)$. The kernels $K_0 = \emptyset$ and K_1 have the same type by convention.

The construction scheme of a CDS may be intuitively explained as follows: Let us suppose that K_i is a contraction kernel and K_{i+1} a removal kernel. The sequence of darts $CW_{i+1}(d) = d.d_1, \ldots, d_p$ connects d to $\sigma_{i+1}(d)$ in G_i (Eq. (4)). Each dart of $CW_{i+1}(d)$ is related to the following one by (Eq. (3)):

$$d_1 = \sigma_i(d),\ \forall j \in \{1, \ldots, p-1\} \quad d_{j+1} = \sigma_i(d_j)$$

Since K_i is a contraction kernel, $CDS_i(d_j)$ connects d_j to $\varphi_i(d_j)$ in G_0 for any j in $\{1, \ldots, p-1\}$. However, $CDS_i(\alpha(d_j))$ connects $\alpha(d_j)$ to $\varphi_i(\alpha(d_j)) = \sigma_i(\alpha \circ \alpha(d_j)) = \sigma_i(d_j)$ (Eq. (1)). The connection between d_j and $\sigma_i(d_j)$ is thus performed by $d_j CDS^*_i(\alpha(d_j))$ and the connecting dart sequence of d at level $i+1$ is equal to $CDS_{i+1}(d) = d_1 \cdot CDS^*_i(\alpha(d_1)) \cdots d_p \cdot CDS^*_i(\alpha(d_p))$.

Note that given a dart $d \in \mathcal{SD}_1$, both $CDS_1(d)$ and $CW_1(d)$ connect d to $\varphi_1(d)$ or $\sigma_1(d)$ according to K_1. Since both sequences are defined in the initial combinatorial map they should be equal. Indeed, one can easily show from Definition 2 that:

$$\forall d \in \mathcal{SD}_1 \quad CDS_1(d) = CW_1(d) \tag{8}$$

5.2 Properties of CDS

Each connecting dart sequence $CDS_i(d)$ defined at level i by $d \in \mathcal{SD}_i$ satisfies [10] $CDS^*_i(d) \subset \cup_{j=1}^{i} K_i$. Therefore, d is the only dart of $CDS_i(d)$ surviving up to level i. Moreover, by construction, connecting dart sequences encode the set of darts in G_0 that we have to traverse to connect one surviving dart at

level i with its φ_i or σ_i successor according to K_i. Indeed, given one dart $d \in \mathcal{SD}_i$, such that $CDS_i(d) = d.d_1, \ldots, d_p$ we have shown [10] that:

$$\pi_i(d) = \begin{cases} \varphi(d_p) \text{ if } d_p \text{ has been contracted} \\ \sigma(d_p) \text{ if } d_p \text{ has been removed} \end{cases} \tag{9}$$

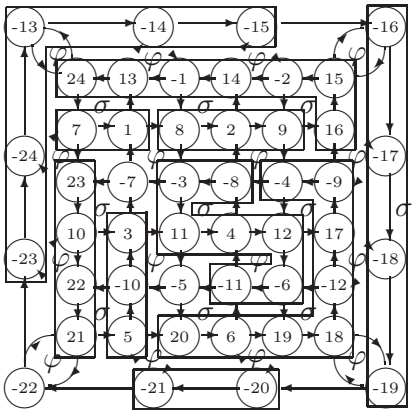
where π_i is associated with φ_i if K_i is a contraction kernel and with σ_i otherwise.

Note that Eq. (9) is similar to Eq. (4) defined for connecting walks. The additional test performed on d_p in Eq. (9) may be intuitively explained by the structure of a connecting dart sequence which contains both contracted and removed darts.

Fig. 11(a) shows the connecting dart sequences defined by the applications of the contraction kernel $K_1 = \alpha^*(1, 2, 4, 12, 6)$ followed by the removal kernel $K_2 = \{\alpha^*(15, 14, 13, 24), \alpha^*(9), \alpha^*(11, 3), \alpha^*(19, 18, 17), \alpha^*(22, 21)\}$(Section 3) superimposed to $\mathcal{OG}$. The construction of $CDS_2(8)$ is detailed in Fig. 11(b).

5.3 The receptive field of a dart

Using the same approach than in Section 4 we define the receptive field of a dart $d \in \mathcal{SD}_i$ as the sequence of darts to traverse in the base level combinatorial map G_0 to connect d to $\sigma_i(d)$. Using Eq. (9) this sequence denoted by



(a) Connecting dart sequences at level 2

Using Fig. 9(b) we have $CW_2(8) = 8.9$. Moreover, using Fig. 9(a), we have $CW_1(-8) = -8.2$ and $CW_1(-9) = -9$. We obtain thus (Definition 2 and Eq. (8)):

$$\begin{aligned} CDS_2(8) &= 8.CDS_1^*(-8).9.CDS_1^*(-9) \\ &= 8.CW_1^*(-8).9.CW_1^*(-9) \\ &= 8.2.9 \end{aligned}$$

(b) Construction of $CDS_2(8)$

Fig. 11. Connecting dart sequences CDS_2 superimposed to $\mathcal{OG}$.

$RF_i(d)$ is equal to:

$$RF_i(d) = \begin{cases} CDS_i(d) & \text{if } K_i \text{ is a removal kernel} \\ d.CDS_i^*(\alpha(d)) & \text{if } K_i \text{ is a contraction kernel.} \end{cases} \quad (10)$$

Let us consider a dart $d \in \mathcal{SD}_i$ such that $RF_i(d) = d.d_1 \dots d_p$. If K_i is a removal kernel, $RF_i(d) = CDS_i(d)$ and $\sigma_i(d)$ is equal to either $\sigma(d_p)$ or $\varphi(d_p)$ according to the operation which suppressed d_p (Eq. (9)). In the same way, if K_i is a contraction kernel, we have $RF_i(d) = d.CDS_i^*(\alpha(d))$ and $\varphi_i(\alpha(d)) = \sigma_i(d)$ is equal to $\sigma(d_p)$ or $\varphi(d_p)$ according to the operation which suppressed d_p (Eq. (9)). We have thus in both cases:

$$\sigma_i(d) = \begin{cases} \varphi(d_p) \text{ if } d_p \text{ has been contracted} \\ \sigma(d_p) \text{ if } d_p \text{ has been removed} \end{cases} \quad (11)$$

where d_p is the last dart of $RF_i(d)$.

The receptive field of $d \in \mathcal{SD}_i$ connects thus the dart d to $\sigma_i(d)$ in G_0. Moreover, the set of receptive fields defined at each level defines a partition of the initial set of darts $\mathcal{D}$ [10].

6 Folding and Unfolding of the Pyramid

Using Eq. (10), the definition of the receptive field of a dart is based on connecting dart sequences. However, using Definition 2 connecting dart sequences must be computed at each level from the level below. Such a construction scheme may induce useless calculus if one does not need to compute connecting dart sequences or receptive fields at all intermediate levels This recursive construction scheme may be avoided by using the following theorem [10]:

Theorem 1 *Given a combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$, a sequence of contraction kernels or removal kernels $K_1, K_2 \dots, K_n$, the relation between the successive darts of a receptive field $RF_i(d) = d.d_1 \dots d_p$, with $i \in \{1, \dots, n\}$ and $d \in \mathcal{SD}_i$ is defined by $d_1 = \sigma(d)$ and:*

$$\forall j \in \{2, \dots, p\}\ d_j = \begin{cases} \varphi(d_{j-1}) \text{ if } d_{j-1} \text{ has been contracted} \\ \sigma(d_{j-1}) \text{ if } d_{j-1} \text{ has been removed} \end{cases}$$

One receptive field defined at level i may thus be traversed if we know the initial dart which defines it and the operation which has reduced each dart of the receptive field. The above remark suggests an encoding of the pyramid by two functions:

(1) Function *state* from $\{1, \ldots, n\}$ to the binary states $\{Contracted, Removed\}$ specifies the type of each kernel.
(2) Function *level* defines for all darts in $\mathcal{D}$ the highest level to which d survives:

$$\forall d \in \mathcal{D}\ level(d) = Max\{i \in \{1, \ldots, n+1\} \mid d \in \mathcal{SD}_{i-1}\} \tag{12}$$

a dart d surviving up to the top level has thus a level equal to $n+1$. Note that the level of an initial vertex (i.e. a pixel) may be implicitly defined as the minimal level of its darts.

Given the function *level*, the sequence of kernels and the set of surviving darts may be retrieved from the base level combinatorial map by the following equations [10]:

$$\forall i \in \{1, \ldots, n\} \begin{cases} K_i &= \{d \in \mathcal{D} \mid level(d) = i\} \\ \mathcal{SD}_i &= \{d \in \mathcal{D} \mid level(d) > i\} \end{cases} \tag{13}$$

Moreover, given a dart $d \in \mathcal{SD}_i$ and using both functions *state* and *level*, the traversal of a receptive field defined by Theorem 1 may be written as follows:

$$\begin{aligned} &d_1 = \sigma(d) \text{ and for each } j \text{ in } \{2, \ldots, p\} \\ &d_j = \begin{cases} \varphi(d_{j-1}) \text{ if } state(level(d_{j-1})) = Contracted \\ \sigma(d_{j-1}) \text{ if } state(level(d_{j-1})) = Removed \end{cases} \end{aligned} \tag{14}$$

Given a receptive field $RF_i(d) = d.d_1 \ldots d_p$, if we define d_{p+1} by $\varphi(d_p)$ if d_p is contracted and by $\sigma(d_p)$ if d_p is removed, then d_{p+1} is equal to $\sigma_i(d)$ (Eq. (11)). We have thus, $d_{p+1} \in \mathcal{SD}_i$ and $level(d_{p+1}) > i$ (Eq. (13)). This last remark allows us to determine the last dart of a receptive field as the one whose successor defined by Eq. (14) has a level strictly greater than i.

The above considerations lead to the design of the function `receptive_field` (Algorithm 1) which computes the receptive field of a dart $d \in \mathcal{SD}_i$. The second dart of the receptive field is initialized on line 4, while the remaining darts of the receptive field are determined according to Eq. (14) (lines 8-11).

Algorithm 1 traverses each dart of the receptive field only once. Since the set of receptive field forms a partition of the initial set of darts $\mathcal{D}$ (Section 5.3),

```
sequence of darts receptive_field(map G_0,dart d,level i)
{
    sequence of darts rf = d;
    dart d' = σ(d);
    while(level(d') ≤ i)
    {
        rf = rf.d';
        if (state(level(d')) == Contracted)
            d' = φ(d');
        else
            d' = σ(d');
    }
    return rf;
}
```

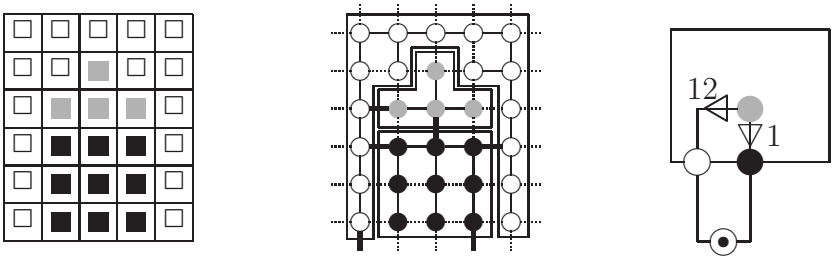
Algorithm 1: *Computation of the receptive field of a dart d at level i.*

the computation of all receptive fields may be performed in $\mathcal{O}(|\mathcal{D}|)$ steps [10] where $|\mathcal{D}|$ denotes the cardinal of $\mathcal{D}$. Note that the reduced combinatorial map at level i may be easily deduced from the set of receptive fields defined at this level (Eq. (11)). We also designed [10] a parallel version of Algorithm 1. This last algorithm traverses all the receptive fields in $\mathcal{O}(\log(|RF_i^{max}(d)|))$ steps where $RF_i^{max}(d)$ denotes the longest receptive field defined at level i.

Since both darts of an edge are simultaneously contracted or removed the function *level* may be initialized on only one dart of each edge according to a particular convention (e.g. the positive dart if α is encoded by the sign). Since the memory requirements of the function *state* is negligible besides the one of function *level* the memory requirements of our encoding is $\frac{|\mathcal{D}|}{2}$ bytes. On the other hand an explicit encoding of the permutation σ for each level requires $|\mathcal{D}|\left(2+\frac{1}{k-1}\right)$ bytes where k is the reduction factor. The ratio between the memory requirements of both encodings is thus at least equal to 4. Note that, in practice this ratio is greater than 4 since additional information are stored in each level of the pyramid.

6.1 Application to Connected Components

Fig. 12 represents an application of the above formalism to the analysis of connected components. The image in Fig. 12(a) is reduced by a pyramid composed of three levels. The initial combinatorial map G_0 encodes the 5×6 4-connected sampling grid (Fig. 12(b)), the combinatorial map G_1 is deduced from G_0 by a contraction kernel K_1 represented by normal lines in Fig. 12(b). The redundant edges are then removed by a removal kernel K_2 represented by dotted lines in Fig. 12(b). The final combinatorial map G_2 encodes the connected



(a) Initial image (b) Folded pyramid in G_0 (c) Top of the pyramid

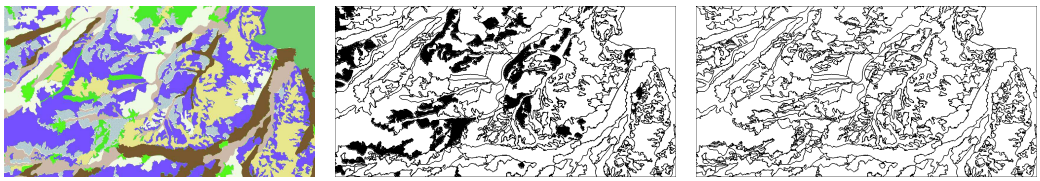
Fig. 12. Folding of a pyramid. The adjacency of the connected components of the initial image (a) are encoded by the top of the pyramid (c). The surviving, contracted and removed edges are respectively represented by thick, normal and dotted lines in figure (b). The vertex with a filled circle ($\odot$) in figure (c) encodes the background of the image.

components of the image and is represented in Fig. 12(c). The corresponding surviving edges are represented in the base level combinatorial map G_0 by thick lines (Fig. 12(b)). The trees composing the contraction and removal kernels may be defined in parallel using the method defined by Montanvert to analyse the connected components of labeled images [3]. Note that since the pyramid is made of only 3 levels, the contracted, removed and surviving edges have a level respectively equal to 1, 2 and 3. The folding of the pyramid defined by the two functions *level* and *state* may thus be read in Fig. 12(b).

The gray roof of the house represented in Fig. 12(a) is encoded by the gray vertex in Fig. 12(c). The σ-orbit of this vertex is composed of the two darts 1 and 12. The receptive fields of these darts are represented in Fig. 13 both in G_0 and $\overline{G_0}$. Using Eq. (14), and the previous convention on the drawing of the darts, the receptive fields may be read on this figure and we have: $RF_2(1)$ =1.2.3.4.5.-2.6.7.8.9.-6.10.11 and $RF_2(12)$ =12.13.-10. Note that the darts 11 and -10 are respectively removed and contracted. Therefore, using Eq. (11) we have: $\sigma_2(1) = \sigma_0(11) = 12$ and $\sigma_2(12) = \varphi_0(-10) = 1$. A similar operation may be applied to all the surviving darts at level 2. The pyramid may thus be unfolded from the base using the functions *level* and *state*. Conversely, the receptive fields of the surviving darts 1 and 12 provide the set of darts of the base level vertices mapped onto the top level vertex $\sigma_2^*(1)$. The set of 0, 1 or 2 cells of the associated region may then be recovered by computing the appropriate orbits in G_0 (see Eq. (2)). Global parameters such as the mean color of the region may be deduced from the values associated to the 2 cells (or pixels) while the exact shape of the region may be deduced from the 0 and 1 cells associated to the external boundary of the receptive field (Section 4)

(a) RF in G_0 (b) RF in $\overline{G_0}$

Fig. 13. The receptive fields of the darts 1 and 12 in Fig 12 represented both in G and $\overline{G}$. The convention on the drawing of the edges are the same as in Fig. 12



(a) Connected components (b) Boundaries (c) Boundaries after the removal of patterns

Fig. 14. Encoding of the connected components of Image(a). Image (b) represents the boundary associated to each edge. Image (c) represents the same boundaries after the removal of small patterns.

Fig. 14 shows another example on a satellite image [2] . The connected components of this image are encoded by a combinatorial pyramid made of 3 levels G_0, G_1 and G_2 using the same method than the one used for Fig. 12. The boundaries of the regions encoded by the combinatorial map G_2 are determined by computing the receptive fields of all darts and selecting the initial vertices whose σ orbit contain a dart belonging to the external boundaries (Section 4). These selected vertices are represented by black pixels in Fig. 14(b). The filled regions in Fig. 14(b) correspond to regions composed of small patterns. The vertices encoding these patterns are associated with regions whose size is smaller than 2. These patterns are removed by merging to one of its neighbor any vertex in G_2 with a size less than 2 pixels. The resulting pyramid is made of 4 levels G_0 to G_4, the boundaries associated to G_4 being displayed in Fig. 14(c). Note that only the top level of the pyramid G_4 is explicitly encoded. The other levels are encoded implicitly thanks to the functions *level* and *state* stored on the base level combinatorial map. We did not represent in Fig. 14(b) and (c) the pseudo-edges encoding inclusion relationships. Such edges correspond to self-loops and are characterized by $\alpha(b) \in \sigma^*(b)$.

[2] Data provided by the Institute of Surveying, Remote Sensing and Land Information, BOKU Vienna

7 Discussion

A combinatorial pyramid consists of a tapering stack of successively reduced combinatorial maps describing the contents of the base level at increasing degrees of abstraction. High level interpretation may need to verify the details of the high level description. The top-down verification can be achieved through the following concepts: A reduction window expands a dart into a sequence of darts in the level below. A receptive field combines the expansion over several levels and over several types of kernels.

Within the simple graph pyramid framework [3] the receptive field of one vertex at level i is determined by iterating the father-child relationship defined by the reduction window from level i to the base level. All intermediate reduced graphs between the base of the pyramid and level i are thus required in order to compute the receptive fields at level i. Kropatsch has shown [14] that within the dual graph pyramid framework, the successive application of several contraction kernels is equivalent to the application of one kernel named equivalent contraction kernel (ECK). A labeling of the contracted edges in the initial dual graph allows Kropatsch to retrieve any contracted graph of the pyramid. However, This framework does not encodes removed edges. The removal kernel associated to each ECK K_i must thus be computed and applied to the contracted graphs $(G_i, \overline{G_i})$.

Our framework allows us to avoid an explicit encoding of all intermediate levels. Moreover, using the function *state* the contraction operations encoded by contraction kernels and the "cleaning" stage performed by removal kernels are encoded into a same framework. The lower space requirements of our encoding may be used to store or to transmit a computed pyramid. However, our encoding may also be used during the construction of the pyramid to retrieve an intermediate level or to compute the receptive field of a dart. The lower memory requirements of our model may be a critical advantage when processing large images or n dimensional data [13].

References

[1] P. Burt, T.-H. Hong, A. Rosenfeld, Segmentation and estimation of image region properties through cooperative hierarchial computation, IEEE Transactions on Sustems, Man and Cybernetics 11 (12) (1981) 802–809.

[2] M. Bister, J. Cornelis, A. Rosenfeld, A critical view of pyramid segmentation algorithms, Pattern Recognit Letter. 11 (9) (1990) 605–617.

[3] A. Montanvert, P. Meer, A. Rosenfeld, Hierarchical image analysis using

irregular tessellations, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (4) (1991) 307–316.

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

[5] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Mesh optimization, in: J. T. Kajiya (Ed.), Computer Graphics (SIGGRAPH'93 Proceedings), volume 27, 1993, pp. 19–26.

[6] J. Weickert (Ed.), Anisotropic Diffusion in Image Processing, Teubner, B.G., 1998.

[7] S. Mallat, Wavelets for a vision, Proceedings of the IEEE 84 (4) (1996) 604–614.
URL `citeseer.nj.nec.com/mallat96wavelets.html`

[8] P. Lienhardt, Topological models for boundary representations: a comparison with n-dimensional generalized maps, Computer-Aided Design 23 (1) (1991) 59–82.

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

[10] L. Brun, W. Kropatsch, The construction of pyramids with combinatorial maps, Tech. Rep. 63, Institute of Computer Aided Design, Vienna University of Technology, Favoritenstr. 9/2/4,A-1040 Vienna Austria (June 2000).
URL `http://www.prip.tuwien.ac.at/`

[11] L. Brun, W. Kropatsch, Contraction kernels and combinatorial maps, in: J. M. Jolion, W. Kropatsch, M. Vento (Eds.), 3^{rd} IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, IAPR-TC15, CUEN, Ischia Italy, 2001, pp. 12–21.

[12] V. Kovalevsky, Finite topology as applied to image analysis, Computer Vision, Graphics, and Image Processing 46 (1989) 141–161.

[13] G. Damiand, P. Lienhardt, Removal and contraction for n-dimensional generalized maps, in: H. Wildenauer, W. Kropatsch (Eds.), Proceedings of the Computer Vision Winter Workshop, Bad Ausse Austria, 2002, pp. 208–221.

[14] W. G. Kropatsch, Equivalent Contraction Kernels and The Domain of Dual Irregular Pyramids, Tech. Rep. PRIP-TR-42, TU Wien, Austria (1995).
URL `www.prip.tuwien.ac.at/ftp/pub/publications/trs/tr42.ps.gz`

[15] L. Brun, W. G. Kropatsch, Dual contraction of combinatorial maps, in: W. G. Kropatsch, J.-M. Jolion (Eds.), 2^{nd} IAPR-TC-15 Workshop on Graph-based Representations, Vol. 126, Österreichische Computer Gesellschaft, Haindorf, Austria, 1999, pp. 145–154.

[16] L. Brun, W. Kropatsch, Defining regions within the combinatorial pyramid framework, in: H. Wildenauer, W. Kropatsch (Eds.), Proceedings of the Computer Vision Winter Workshop, Bad Ausse Austria, 2002, pp. 198–207.