Canonical Encoding of the Combinatorial Pyramid

Fuensanta Torres and Walter G. Kropatsch

PRIP, Vienna University of Technology, Austria
{fuensanta, krw}@prip.tuwien.ac.at

Abstract *This paper presents a novel framework to encode a combinatorial pyramid. A combinatorial pyramid is a hierarchy of successively reduced combinatorial maps. Important properties of the combinatorial pyramids such as topology preservation, the process global and local features within the same data structure, etc. made them useful for image processing and pattern recognition tasks. Their advantages have been widely proved in the literature. Nevertheless, the main disadvantage of this approach is the high rate of memory requirement. A combinatorial map of an image maybe stored in an array of size approximately equal to four times the number of pixels of the image. Furthermore, every level of the combinatorial pyramid stores a different combinatorial map. In respond to this problem a canonical encoding of the combinatorial pyramid is provided. It consists of a single array where its elements are ordered with respect to the construction history of the pyramid. In this manner the memory consumptions are equal to the size of the initial combinatorial map and do not depend on the number of pyramid's levels. In addition, this canonical encoding allows the whole reconstruction of the pyramid in both directions: from the base to the top level and from the top to the base level, without additional information.*

1 Introduction

A 2D combinatorial map [11, 15] defines a data structure to encode the subdivision of the plane in different regions. It has more advantages compared to the traditional Region Adjacency Graphs (RAG):

- The combinatorial maps represent topological information (multi-adjacency or inclusion relations). Unlike the RAG, where two topologically different images could be represented by the same RAG.
- They can be extended to higher dimensions (nD).
- They allow efficient algorithms to retrieve information and modify the partition.

The combinatorial pyramid [4] is a stack of successively reduced combinatorial maps. Such structure takes advantages of the combinatorial maps as well as benefits from additional properties:

- The combinatorial pyramids preserve topology.
- They process global and local features within the same data structure.

Nowadays, the combinatorial maps and the combinatorial pyramids are applied for various tasks such as image segmentation [1, 8], map matching [9, 14, 13, 17], 3D mesh representation [10], etc. These structures require high memory consumptions, and this requirement is exacerbated by the pyramid -the pyramid structure stores one combinatorial map at each level of its hierarchy.
In the literature exits several methods to reduce the memory consumptions [12, 16, 5].
Goffe et al [12] segment the initial image -they pre-process the initial image. They find the combinatorial map for this segmented image -it is the top level of the pyramid. And they progressively create the lower levels of the hierarchy by increasing the size of the combinatorial maps.
[16] and [5] define the implicit encoding of the 2-dimensional and n-dimensional combinatorial pyramid, respectively. For the 2D implicit encoding, Brun et al [5] store the combinatorial map of the initial image and two functions which describe the construction history -one function specifies the type of operation done over each element of the initial combinatorial map and the other function defines the highest level of the pyramid until which the element survives.

The canonical encoding of the combinatorial pyramid stores the whole pyramid and its construction history in the same memory than the combinatorial map of the initial image. The operation applied to each element of the initial combinatorial map is implicitly encoded in the representation; and the highest level until which the element survive are implicitly encoded in the order of the elements. In this manner the memory consumptions are equal to the initial combinatorial map and do not depend on the number of levels of the pyramid, as previous works. It allows the construction of the pyramid from the top to the base level without additional information. Meanwhile Goffe et al [12] need additional information (parent/child relations).

The remaining of this paper is organized as follows: Basic definitions on combinatorial maps and combinatorial pyramids are recalled in sections 2 and 3. Section 4 describes our contribution -the canonical encoding of the combinatorial pyramid. The experimental results proving the theoretical concept of the canonical encoding are described in Section 5. Finally, conclusions and future work are un-

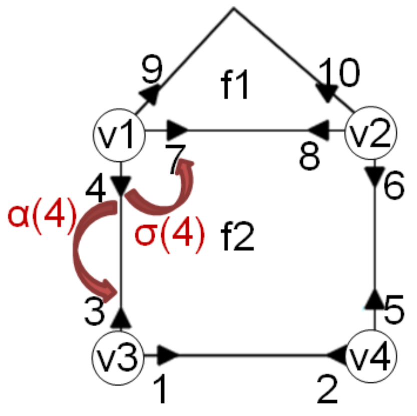

Figure 1: Combinatorial map example.

Table 1: Combinatorial Map (G=($\mathcal{D}$,α,σ)) of Fig. 1.

darts (d)	1	2	3	4	5	6	7	8	9	10
α	2	1	4	3	6	5	8	7	10	9
σ	3	5	1	7	2	10	9	6	4	8

folded at Section 6.

2 Recalls on Combinatorial Maps

Definition 1. ***2D Combinatorial Map:*** *A 2D combinatorial map encodes the subdivision of the plane in different regions. It represents the inclusion and adjacency relations between the regions. This principle enables to fully describe the topology of the plane partition [11, 15]. The 2D combinatorial map (G) is defined by a triplet G=($\mathcal{D}$, α, σ), where:*

- *$\mathcal{D}$ is a finite set of darts.*
- *α is an involution on the set $\mathcal{D}$.*
- *σ is a permutation on the set $\mathcal{D}$.*

An additional explanations of the definition above: The edges (paths connecting two vertices) are divided into two half edges called darts. α is an one-to-one mapping between consecutive darts forming the same edge, such that $\alpha(\alpha$(d))=d). σ is a mapping between consecutive darts around the same vertex while turning counterclockwise.

Example. 2D Combinatorial Map: Fig. 1 and Tab. 1 give an example of a 2D combinatorial map. Where:

- $\mathcal{D} = \{$ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 $\}$.
- $\alpha(4) = 3$.
- $\sigma(4) = 7$.

3 Recalls on Combinatorial Pyramids

Definition 2. ***Combinatorial Pyramid:*** *A combinatorial pyramid is a stack of successively reduced combinatorial maps (Fig. 2). The size of the combinatorial maps is successively reduced by contractions and removals. The contraction and the removal kernels specify a set of darts, which will be contracted and removed, respectively, between two consecutive levels (as previously described [7]). Given an initial combinatorial map G_0=($\mathcal{D}$,α,σ) and the sequence of kernels (k_1, k_2, k_3,..., k_n) we can build the stack of successively reduced combinatorial maps G_1, G_2,..., G_n. The combinatorial maps at all the levels preserve the topology of the initial combinatorial map [3, 6].*

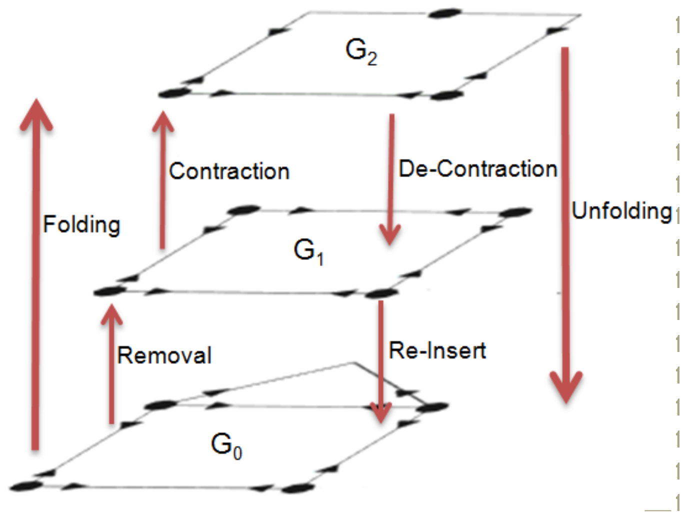

Figure 2: Combinatorial pyramid example.

Example. Combinatorial Pyramid: Fig. 2 gives an example of a combinatorial pyramid. Where:

- G_0 (the base level in the pyramid) is equal to the combinatorial map of Fig. 1.
- We reduce the number of darts in G_0 by the removal kernel k_1 = {9, 10} and we obtain G_1.
- We apply on G_1 the contraction kernel k_2 = {7, 8} and we get the top level (G_2) of the pyramid.

4 Folding and Unfolding the Pyramid

The aim of this section is to explain the canonical encoding of the combinatorial pyramid.

4.1 Folding the Pyramid

The operations used to build a pyramid are the removal and contraction operations. The kernels K (as described in Sec. 3) selects the darts which will be removed or contracted. The removal and contraction involve 6 dependent darts (Fig. 3): d, $\alpha(d)$, f= $\sigma^{-1}(d)$, g= $\sigma(d)$, h= $\sigma^{-1}(\alpha(d))$, i= $\sigma(\alpha(d))$. Special cases are the empty self-loop and the pending edge where $\sigma(d) = \alpha(d)$ and $\sigma(\alpha(d)) = \alpha(d)$, respectively.
We need additional definitions in order to be able to define the contraction and removal operations [2]:

Definition 3. *σ^* is the σ orbit, which defines all the darts belonging to the same vertex.*

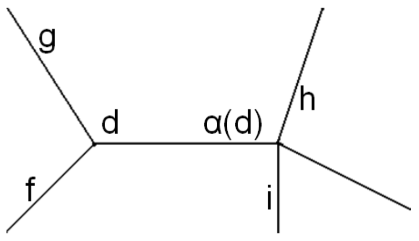

Figure 3: Dependent Darts in the operations.

NOTE: $\sigma^{-1}(d)= \sigma^{n}(d)$ with n= $\min\{i \mid \sigma^{i}(d) = d\}$; where n is the number of successive applications of σ on the dart d.

Definition 4. *α^* is the α orbit, which defines the pair of darts belonging to the same edge.*

Definition 5. ***Empty Self loop:*** *A pair of darts are an empty self loop, iff $\sigma(d) = \alpha(d)$ for $d \in \alpha^*$.*

Definition 6. ***Pending edge:*** *A pair of darts are a pending edge, iff $\sigma(\alpha(d)) = \alpha(d)$ for $d \in \alpha^*$.*

Definition 7. ***Removal Operation [2]:*** *We remove a pair of darts $\alpha^*(d)$ from G=($\mathcal{D}$,α,σ), $\mathcal{D}'$= $\mathcal{D}$ $\setminus\alpha^*(d)$. And in addition, we modify the permutation σ and we obtain σ'. The values of σ' for the all the d' $\in \mathcal{D}'$ are:*

- *if $\alpha^*(d)$ is not an empty self loop ($\sigma(d) = \alpha(d)$) and neither a pending edge ($\sigma(\alpha(d)) = \alpha(d)$):*

$$\sigma'(\sigma^{-1}(d)) = \sigma(d) \tag{1}$$

$$\sigma'(\sigma^{-1}(\alpha(d))) = \sigma(\alpha(d)) \tag{2}$$

- *if $\alpha^*(d)$ is an empty self loop ($\sigma(d) = \alpha(d)$):*

$$\sigma'(\sigma^{-1}(d)) = \sigma(\alpha(d)) \tag{3}$$

- *For the rest of darts, which are not included in the cases above:*

$$\forall d' \in D \setminus \{\sigma^{-1}(d), \sigma^{-1}(\alpha(d))\}\sigma'(d') = \sigma(d') \tag{4}$$

Parts disappearing from G are saved as left over of the removal (LoR). LoR is a quadruplet $\{p, q, r, s\} \in \mathcal{D}^{\,4}$, where p= d, q= $\alpha(d)$ r=$\sigma(d)$ and s=$\sigma(\alpha(d))$.

Proposition 1.

1. *The triplet ($\mathcal{D}'$, α, σ') form a valid combinatorial map(G').*
2. *In an empty self-loop:*

$$r = q \tag{5}$$

3. *In a pending edge:*

$$s = q \tag{6}$$

Proof. (1) The proof that G' is a valid combinatorial map without $\alpha^*(d)$ can be found in [2] □

Proof. (2) $r = \sigma(d)$, $q = \alpha(d) \Rightarrow \sigma(d)$= $\alpha(d)$ is an empty self-loop. □

Proof. (3) $s = \sigma(\alpha(d))$, $q = \alpha(d) \Rightarrow \sigma(\alpha(d))$= $\alpha(d)$ is a pending edge. □

Definition 8. ***Contraction Operation [2]:***
We remove a pair of darts $\alpha^*(d)$ from G=($\mathcal{D}$,α,σ), $\mathcal{D}'$ = $\mathcal{D}$ $\setminus\alpha^*(d)$. And in addition, we modify the permutation σ and we obtain σ'. The values of σ' for the all the d' $\in \mathcal{D}'$ are:

- *if $\alpha^*(d)$ is not an empty self loop ($\sigma(d) = \alpha(d)$) and neither a pending edge ($\sigma(\alpha(d)) = \alpha(d)$):*

$$\sigma'(\sigma^{-1}(d)) = \sigma(\alpha(d)) \tag{7}$$

$$\sigma'(\sigma^{-1}(\alpha(d))) = \sigma(d) \tag{8}$$

- *if $\alpha^*(d)$ is a pending edge ($\sigma(\alpha(d)) = \alpha(d)$):*

$$\sigma'(\sigma^{-1}(d)) = \sigma(d) \tag{9}$$

- *For the rest of darts, which are not included in the cases above:*

$$\forall d' \in D \setminus \{\sigma^{-1}(d), \sigma^{-1}(\alpha(d))\}\sigma'(d') = \sigma(d') \tag{10}$$

Parts disappearing from G are saved as left over of the contraction (LoC). LoC is a quadruplet $\{p, q, r, s\} \in \mathcal{D}^{\,4}$, where p= d, q= $\alpha(d)$ r=$\sigma(d)$ and s=$\sigma(\alpha(d))$.

Proposition 2.

1. *The triplet ($\mathcal{D}'$, α, σ') form a valid combinatorial map(G').*
2. *In an empty self-loop:*

$$r = q \tag{11}$$

3. *In a pending edge:*

$$s = q \tag{12}$$

Proof. (1) The proof that G' is a valid combinatorial map without $\alpha^*(d)$ can be found in [2] □

Proof. (2) $r = \sigma(d)$, $q = \alpha(d) \Rightarrow \sigma(d)$= $\alpha(d)$ is an empty self-loop. □

Proof. (3) $s = \sigma(\alpha(d))$, $q = \alpha(d) \Rightarrow \sigma(\alpha(d))$= $\alpha(d)$ is a pending edge. □

Tab. 2 summarize the contraction and removal operations.

Example. Remove Empty Self-Loop: Fig. 4 gives an example of an empty self-loop removal. It shows the combinatorial map before the removal (Fig. 4 on the left) and after the removal (Fig. 4 on the right). Where we can see that the value of $\sigma'(3)$ changes according to Tab. 2, $\sigma'(\sigma^{-1}(d)) := \sigma(\alpha(d))$ (eq. 3) (σ(3)=d, σ'(3)=1).

Table 2: Removal and Contraction operations.

case	REDUCE $(d, \alpha(d))$
pending edge	if $\sigma(\alpha(d)) = \alpha(d)$ then $\sigma'(\sigma^{-1}(d)) := \sigma(d)$ (eq. 9)
empty-self-loop	if $\sigma(d) = \alpha(d)$ then $\sigma'(\sigma^{-1}(d)) := \sigma(\alpha(d))$ (eq. 3)
remove	$\sigma'(\sigma^{-1}(d)) := \sigma(d)$ (eq. 1) $\sigma'(\sigma^{-1}(\alpha(d))) := \sigma(\alpha(d))$ (eq. 2)
contract	$\sigma'(\sigma^{-1}(d)) := \sigma(\alpha(d))$ (eq. 7) $\sigma'(\sigma^{-1}(\alpha(d))) := \sigma(d)$ (eq. 8)

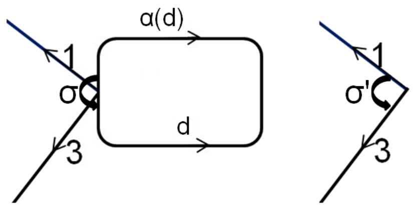

Figure 4: Remove Empty-Self-Loop(d,$\alpha(d)$).

Definition 9. ***The canonical encoding**: The canonical encoding of the combinatorial pyramid is an ordered sequence of darts which fully encodes the combinatorial map G at any level and its construction history. The darts are encoded by even and odd integers; in a way that the involution α could be implicitly encoded by eq. 13.*

$$\alpha(d) = \begin{cases} d+1 & \text{if } d \text{ odd} \\ d-1 & \text{if } d \text{ even} \end{cases} \tag{13}$$

The pyramid is built by a sequence of contraction and removal operations. Each one producing a smaller combinatorial map and the quadruplet of the Lo (Left over). The canonical encoding reorder the darts of the Lo in the chronological order. The surviving darts constitute the active part of the canonical encoding. Meanwhile, the ordered Lo constitute the passive part.

We assume that we have executed n operations in total, then we have n Lo (Tab. 3). The Π function (Π: $\mathcal{D} \to \mathcal{E}$) (eq. 14, 15) gives the new order of this darts in the passive part of the canonical encoding. The unique identifier of each p_t and q_t is assigned to its position in the passive part, therefore we only need to store $\Pi\ (r_t)$ and $\Pi\ (s_t)$ (such that $\mathcal{E}_1$=$\Pi\ (r_1)$, $\mathcal{E}_2$=$\Pi\ (s_1)$, etc.).

$$\Pi(p_t) = 2t - 1 \tag{14}$$

$$\Pi(q_t) = 2t \tag{15}$$

Table 3: Left over table.

t	Left over (Lo)
1	p_1, q_1, r_1, s_1
2	p_2, q_2, r_2, s_2
3	p_3, q_3, r_3, s_3
4	p_4, q_4, r_4, s_4
n	p_n, q_n, r_n, s_n

costruzione piramide

Property. Canonical Encoding (Folding the pyramid): The canonical encoding of the combinatorial pyramid encodes the whole pyramid with the same memory as in the base level -it does not increase with the height of the pyramid. Traditional methods (also called explicit encoding) store the initial combinatorial map at the base level; but they also need additional memory every new level -they store a new combinatorial map per level. The implicit encoding [5] also needs additional memory every new level to describe the construction history of the pyramid.

4.2 Unfolding the Pyramid

In order to retrieve the initial combinatorial map (G_0) from the combinatorial map at any level, the darts in the passive part are moving to the active part. In the folding previous to the unfolding operation, the darts have been ordered with respect to the construction history. Therefore, we only have to shift the boundary between the active and the passive part; and to apply -re-insertion or de-contraction.

Definition 10. ***Re-insertion Operation:** We add the LoR(t)={p, q, r, s} to G'=($\mathcal{D}'$,α,σ'), $\mathcal{D}''$= $\mathcal{D}'$ $\cup$ {p, q}. And in addition, we modify the permutation σ' and we obtain σ''. The values of σ'' for the all the d" $\in \mathcal{D}''$ are:*

- *if $r \neq q$(eq. 5) and $s \neq q$ (eq. 6):*

$$\sigma''(\sigma'^{-1}(r)) := p \tag{16}$$

$$\sigma''(\sigma'^{-1}(s)) := q \tag{17}$$

- *if $r = q$ (eq. 5):*

$$\sigma''(\sigma'^{-1}(s)) := p \tag{18}$$

- *For the rest of darts, which are not included in the cases above:*

$$\forall d'' \in D'' \setminus \{\sigma'^{-1}(p), \sigma'^{-1}(q)\}\sigma''(d'') = \sigma'(d'') \tag{19}$$

The values of α'' for the all the d" $\in \mathcal{D}''$ are:

$$\forall d'' \in D'' \setminus \{p, q\}\alpha''(d'') = \alpha'(d'') \tag{20}$$

$$\{p, q\} \in LoR(t)''\alpha''(p) = q; \alpha''(q) = p \tag{21}$$

Proposition 3.

1. *The triplet ($\mathcal{D}''$, α'', σ'') form a valid combinatorial map(G").*
2. *G"(Def. 10)= G (Def. 7).*

Proof. (1)

- $\mathcal{D}''$= $\mathcal{D}'$ $\cup${p, q} is a finite set of darts.
- $\forall d'' \in \mathcal{D}' \Rightarrow \alpha''$ (d'')= α' (d'') (eq. 20) is an involution.
- $\forall d'' \notin \mathcal{D}' \Rightarrow d'' = p \in LoR$ and α'' $(d'') = q \in LoR$ $\Rightarrow q = \alpha''(p)$ (eq. 21) is an involution.

- $\forall d''$ such that $\sigma'(d'') \neq r$ and $\sigma'(d'') \neq s \Rightarrow \sigma''\ (d'')= \sigma'\ (d'')$ (eq. 19) is a permutation.
- if $\sigma'(d'') = r$ and $r \neq q$ (eq. 5) and $s \neq q$ (eq. 6) $\Rightarrow \sigma''(d'') = p$ (eq. 16) and $\sigma''(p) = r$ (Def. 7) is a permutation.
- if $\sigma'(d'') = s$ and $r \neq q$ (eq. 5) and $s \neq q$ (eq. 6) $\Rightarrow \sigma''(d'') = q$ (eq. 17) and $\sigma''(q) = s$ (Def. 7) is a permutation.
- if $\sigma'(d'') = s$ and $r = q$ (eq. 5) $\Rightarrow \sigma''(d'') = p$ (eq. 18) and $\sigma''(q) = s$ (Def. 7) is a permutation.

□

Proof. (2)

- if $r \neq q$ (eq. 5) and $s \neq q$ (eq. 6):

  Given $\sigma''(\sigma'^{-1}(r)) := p$ (eq. 16). We have $r = \sigma'(\sigma^{-1}(p))$ (eq. 1) thus:
  $\sigma''(\sigma'^{-1}(\sigma'(\sigma^{-1}(d))) = \sigma''((\sigma^{-1}(p))) := p$

  Given $\sigma''(\sigma'^{-1}(s)) := q$ (eq. 17). We have $s = \sigma'(\sigma^{-1}(q))$ (eq. 2) thus:
  $\sigma''(\sigma'^{-1}(\sigma'(\sigma^{-1}(q)))) = \sigma''(\sigma^{-1}(q)) := q$

- if $r = q$ (eq. 5):

  Given $\sigma''(\sigma'^{-1}(s)) := p$ (eq. 18). We have $s = \sigma'(\sigma^{-1}(p))$ (eq. 3) thus:
  $\sigma''(\sigma'^{-1}(\sigma'(\sigma^{-1}(p)))) = \sigma''(\sigma^{-1}(p)) := p$

- $\forall d'' \in \mathcal{D}'' \setminus\{\sigma'^{-1}(r), \sigma'^{-1}(s)\}\ \sigma''(d'') = \sigma'(d'')$ (eq. 19). We have $\forall d' \in \mathcal{D} \setminus\{\sigma^{-1}(d), \sigma^{-1}(\alpha(d))\}\ \sigma'(d') = \sigma(d')$ (eq. 4) thus:
  $\sigma''(d'') = \sigma(d'')$

□

Definition 11. ***De-contraction Operation:*** *We add the LoC(t)=$\{p, q, r, s\}$ to G'=($\mathcal{D}'$,α, σ'), D''= $\mathcal{D}' \cup \{p, q\}$. And in addition, we modify the permutation σ' and we obtain σ''. The values of σ'' for the all the d" $\in \mathcal{D}''$ are:*

- *if $r \neq q$ (eq. 11) and $s \neq q$ (eq. 12):*

$$\sigma''(\sigma'^{-1}(r)) := q \tag{22}$$

$$\sigma''(\sigma'^{-1}(s)) := p \tag{23}$$

- *if $s = q$ (eq. 12):*

$$\sigma''(\sigma'^{-1}(r)) := p \tag{24}$$

- *For the rest of darts, which are not included in the cases above:*

$$\forall d'' \in D \setminus \{\sigma'^{-1}(p), \sigma'^{-1}(q)\}\sigma''(d'') = \sigma'(d'') \tag{25}$$

The values of α'' for the all the d" $\in \mathcal{D}''$ are:

$$\forall d'' \in D'' \setminus \{p, q\}\alpha''(d'') = \alpha'(d'') \tag{26}$$

$$\{p, q\} \in LoC(t)''\alpha''(p) = q; \alpha''(q) = p \tag{27}$$

Proposition 4.

1. *The triplet ($\mathcal{D}''$, α'', σ'') form a valid combinatorial map(G").*
2. *G"(Def. 11)= G (Def. 8).*

Proof. (1)

- $\mathcal{D}''$= $\mathcal{D}'$ $\cup\{p, q\}$ is a finite set of darts.
- $\forall d'' \in \mathcal{D}' \Rightarrow \alpha''\ (d'')$= $\alpha'\ (d'')$ (eq. 26) is an involution.
- $\forall d'' \notin \mathcal{D}' \Rightarrow d'' = p \in LoC$ and $\alpha''\ (d'') = q \in LoC \Rightarrow q = \alpha''(p)$ (eq. 27) is an involution.
- $\forall d''$ such that $\sigma'(d'') \neq r$ and $\sigma'(d'') \neq s \Rightarrow \sigma''\ (d'')= \sigma'\ (d'')$ (eq. 25) is a permutation.
- if $\sigma'(d'') = r$ and $r \neq q$ (eq. 11) and $s \neq q$ (eq. 12) $\Rightarrow \sigma''(d'') = p$ (eq. 23) and $\sigma''(p) = r$ (Def. 8) is a permutation.
- if $\sigma'(d'') = s$ and $r \neq q$ (eq. 11) and $s \neq q$ (eq. 12) $\Rightarrow \sigma''(d'') = q$ (eq. 22) and $\sigma''(q) = s$ (Def. 8) is a permutation.
- if $\sigma'(d'') = r$ and $s = q$ (eq. 12) $\Rightarrow \sigma''(d'') = p$ (eq. 24) and $\sigma''(q) = s$ (Def. 8) is a permutation.

□

Proof. (2)

- if $r \neq q$ (eq. 11) and $s \neq q$ (eq. 12):

  Given $\sigma''(\sigma'^{-1}(r)) := q$ (eq. 22). We have $r = \sigma'(\sigma^{-1}(q))$ (eq. 8) thus:
  $\sigma''(\sigma'^{-1}(\sigma'(\sigma^{-1}(q)))) = \sigma''(\sigma^{-1}(q)) := q$

  Given $\sigma''(\sigma'^{-1}(s)) := p$ (eq. 23). We have $s = \sigma'(\sigma^{-1}(p))$ (eq. 7) thus:
  $\sigma''(\sigma'^{-1}(\sigma'(\sigma^{-1}(p)))) = \sigma''(\sigma^{-1}(p)) := p$

- if $s = q$ (eq. 12):

  Given $\sigma''(\sigma'^{-1}(r)) := p$ (eq. 24). We have $\sigma'(\sigma^{-1}(p)) = r$ (eq. 9).
  $\sigma''(\sigma'^{-1}(\sigma'(\sigma^{-1}(p)))) = \sigma''(\sigma^{-1}(p)) := p$

- $\forall d'' \in \mathcal{D} \setminus\{\sigma'^{-1}(d), \sigma'^{-1}(\alpha(d))\}\ \sigma''(d'') = \sigma'(d'')$ (eq. 25). We have $\forall d' \in \mathcal{D} \setminus\{\sigma^{-1}(d), \sigma^{-1}(\alpha(d))\}$ $\sigma'(d') = \sigma(d')$ (eq. 10) thus:
  $\sigma''(d'') = \sigma(d'')$

Table 4: Re-insert and De-contract operations.

case	EXPAND (p, q)
	if s = q then $\sigma''(\sigma'^{-1}(r)) := p$ (eq. 24)
	if r = q then $\sigma''(\sigma'^{-1}(s)) := p$ (eq. 18)
re-insert -conditions	if $\sigma'(q) \notin \sigma'^{*}(\sigma'(p))$ or $\sigma'(p) \notin \sigma'^{*}(\sigma'(q))$ if $\sigma'^{*}(p) = \sigma'^{*}(q)$
-operations	$\sigma''(\sigma'^{-1}(s)) := q$ (eq. 16) $\sigma''(\sigma'^{-1}(s)) := q$ (eq. 17)
de-contract -conditions	if $\sigma'(q) \in \sigma'^{*}(\sigma'(p))$ or $\sigma'(p) \in \sigma'^{*}(\sigma'(q))$
-operations	$\sigma''(\sigma'^{-1}(r)) := q$ (eq. 22) $\sigma''(\sigma'^{-1}(s)) := p$ (eq. 23)

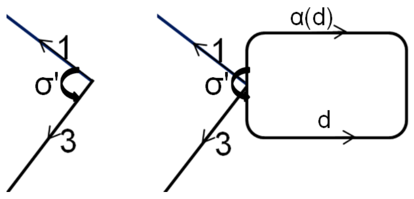

Figure 5: Re-Insert Empty-Self-Loop(d, $\alpha(d)$).

□

Tab. 4 summarizes the re-insertions and de-contractions. It gives the conditions to recognize whether the pair of darts were previously either removed (re-insert conditions) or contracted (de-contract conditions) in the folding procedure.

Example. Re-Insert Empty-Self-Loop: Fig. 5 gives an example of an empty self-loop re-insertion. It shows the combinatorial map before the re-insertion (Fig. 5 on the left) and after the re-insertion (Fig. 5 on the right). Where we can see that the value of $\sigma''(3)$ changes according to Tab. 4, $\sigma''(\sigma'^{-1}(\sigma'(\alpha(d)))) := d$ (eq. 18) ($\sigma'(3) = 1$, $\sigma''(3) = d$).

ricostruire il primo livello

Property. Canonical Encoding (Unfolding the pyramid): Given the canonical encoding of the combinatorial pyramid at any level, the initial combinatorial map (G_0) can be retrieved -without extra information. Traditional methods store the parent/child relations to be able to unfold the pyramid. In the canonical encoding, we detect if we should either re-insert or de-contract the pair of darts, of the passive part, according with Tab. 4. And we apply the corresponding operation either re-insertion or de-contraction.

5 Proof of concepts

Application(Connected component labeling). The canonical encoding of the combinatorial pyramid has been used for connected components labeling. At the base level G_0, a pair of darts (d, α(d)) connects a pixel of the initial image with its 4-neighbors. Each dart stores the color of its related pixel. In the connected component application,

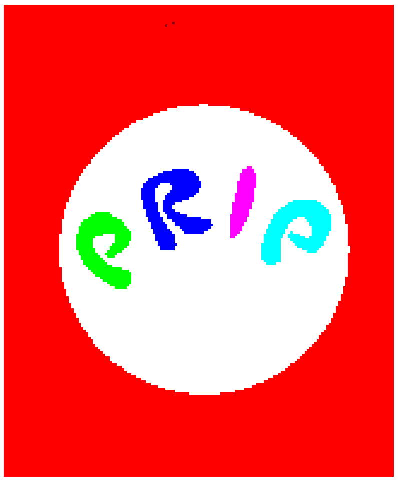

Figure 6: Input Image.

the contraction kernels are composed of darts all having the same color. The contraction may create redundant edges, which constitutes the removal kernels.

Example(Connected components labeling). Fig. 6 is the input image of the combinatorial pyramid. Fig. 7 shows the first level of the pyramid, where the pixels which have all their related darts in the passive part have black color. Fig. 8 shows the combinatorial map at the top level of the pyramid. Each connected component is contracted to a single vertex. The combinatorial map encodes the inclusion and adjacency relations among these connected components. It fully describes the topology of the plane partition.

Property(Memory requirements). A combinatorial map with $|\mathcal{D}|$ darts maybe stored in one array of dimensions equal to $|\mathcal{D}| \times \log_2(\mathcal{D})$ bits (assuming that the unique identifier of each dart (d) is assigned to its position in the array, we only need to store $\sigma(d)$). If the reduction factor between any two levels of the pyramid is K. The number of darts at one level of the pyramid G_l is $\frac{|\mathcal{D}|}{k^l}$. Therefore, the bits used to store the pyramid explicitly is $\log_2(D) \sum_{l=0}^{n} \frac{|\mathcal{D}|}{k^l}$. It also needs in addition the parent/child relations to unfold the pyramid. The implicit encoding [16, 5] requires the storage of the combinatorial map at the base level and one integer for each pair darts. The bits used to store the implicit encoding is $|\mathcal{D}| \log_2(D) + \frac{1}{2}\ |D|\ (\log_2(n))$. The canonical implementation requires only the storage of the combinatorial map at the base level $|\mathcal{D}| \times \log_2(D)$ bits.

6 Conclusions and Future work

In the present work, we propose a canonical encoding of a combinatorial pyramid. This new structure stores the whole combinatorial pyramid and its construction history in the same memory as the initial combinatorial map. We use the order of the darts to encode the information about the pyramid structure. It allows the full reconstruction of the pyra-

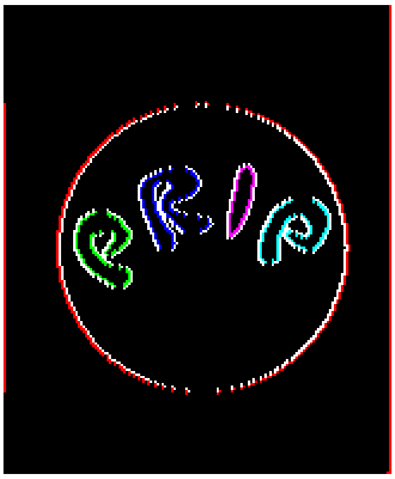

Figure 7: Darts at the first level(D_1).

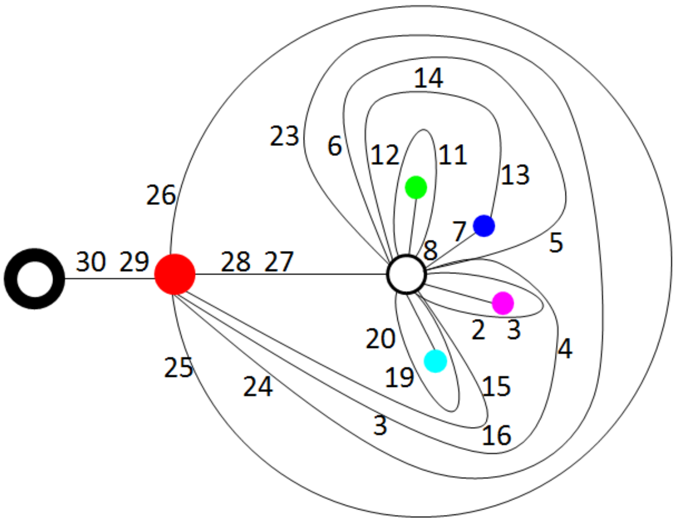

Figure 8: Combinatorial Map at the top level(G_9).

mid in both directions (folding and unfolding the pyramid). Different ways to construct a combinatorial pyramid can be found in the literature. Such methods either store a stack of successively reduced maps; or they store the initial combinatorial map and additional information to describe the construction history of the pyramid. The canonical encoding of the combinatorial pyramid reduces the memory requirements of the previous works without loss the functionality. In our proposed framework to encode the combinatorial pyramid the darts ranked according to its importance on the initial image (i.e. most of the darts of the uniform regions are contracted at the firsts level of the pyramid. They will be in the lasts positions of the canonical encoding). Now, we plan to study this property of our canonical encoding and to build signatures for image matching applications such as in [14, 13].

Acknowledgement

The first author thanks to Doctoral College on Computational Perception (Vienna University of Technology, Austria) for funding.

References

[1] E. Antunez, R. Marfil, J. P. Bandera, and A. Bandera. Part-based object detection into a hierarchy of image segmentations combining color and topology. *Pattern Recognition Letters*, 2013.

[2] L. Brun and W. Kropatsch. Dual contraction of combinatorial maps. Technical report, Pattern Recognition and Image Processing Group. Vienna University of Technology, 1999.

[3] L. Brun and W. G. Kropatsch. Dual contraction of combinatorial maps. *Workshop on Graph-based Representations*, 1999.

[4] L. Brun and W. G. Kropatsch. Introduction to combinatorial pyramids. *Lecture Notes in Computer Science 2243.*, 2001.

[5] L Brun and W. G. Kropatsch. Combinatorial pyramids. *International Conference on Image Processing (ICIP), Barcelona, Spain*, 2003.

[6] L. Brun and W. G. Kropatsch. Construction of combinatorial pyramids. *Hancock, E.R., Vento,M. (eds.) GbRPR 2003. LNCS, vol. 2726, pp. 112. Springer*, 2003.

[7] L. Brun and W. G. Kropatsch. Contraction kernels and combinatorial maps. *Pattern Recognition Letters*, 2003.

[8] L. Brun, M. Mokhtari, and F. Meyer. Hierarchical watersheds within the combinatorial pyramid framework. *Discrete Geometry for Computer Imagery. Springer Berlin Heidelberg*, 2005.

[9] L. Brun and J. H. Pruvot. Hierarchical matching using combinatorial pyramid framework. *Lecture Notes in Computer Science, vol. 5099. Springer, pp. 346355*, 2008.

[10] X. Feng, Y. Wang, Y. Weng, and Y. Tong. Compact combinatorial maps in 3d. *Computational Visual Media. Springer Berlin*, 2012.

[11] A. J. Gareth and D. Singerman. Theory of maps on orientable surfaces. *Proceedings of the London Mathematical Society*, 1978.

[12] R. Goffe, L. Brun, and G. Damiand. Tiled topdown combinatorial pyramids for large images representation. *International Journal of Imaging Systems and Technology*, 2011.

[13] S. Gosselin, G. Damiand, and C. Solnon. Efficient search of combinatorial maps. *Unknown Journal*, 2011.

[14] S. Gosselin, G. Damiand, and C. Solnon. Frequent submap discovery. *Combinatorial Pattern Matching, Springer Berlin Heidelberg*, 2011.

[15] P. Lienhardt. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer-Aided Design 23(1)*, 1991.

[16] F. Sbastien and Luc Brun. Efficient encoding of nd combinatorial pyramids. *International Conference on Pattern Recognition (ICPR)*, 2010.

[17] T. Wang, G. Dai, B. Ni, D. Xu, and F. Siewe. A distance measure between labeled combinatorial maps. *Computer Vision and Image Understanding*, 116(12):1168 – 1177, 2012.