Partitions et structures hiérarchiques

Luc Brun

Groupe de Recherche en Informatique,
Image, Automatique et Instrumentation
de Caen (GREYC)

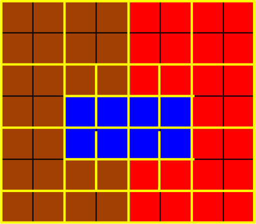
Plan

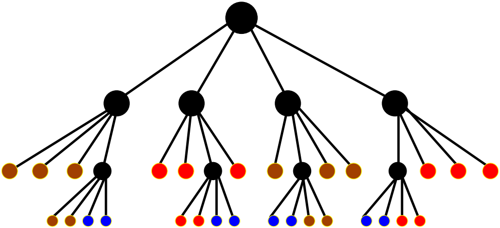
Introduction

- Defining a partition involves a choice :

- All usefull information must be in the provided partition ☹
- Provides not one but a full stack of partitions successively reduced. ☺

T-pyramids (quadtrees)

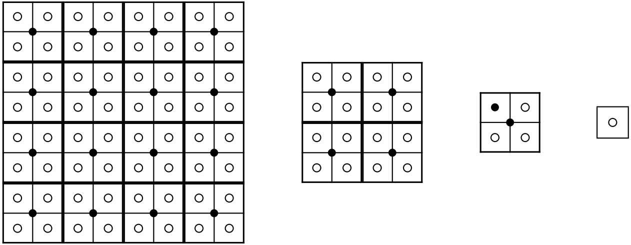




- Top-down construction of the partition by a recursive decomposition into squares.
- Avantages :
  - Efficient Acces to some geometrical information.
- Drawbacks (see M-pyramids below)

Matrix Pyramids

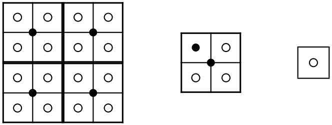
- Stack of images with progressively reduced resolution.



A $2 \times 2/4$ pyramid

- At each step the pixels of the image above (□) may be associated to pointels (•)of the image below.

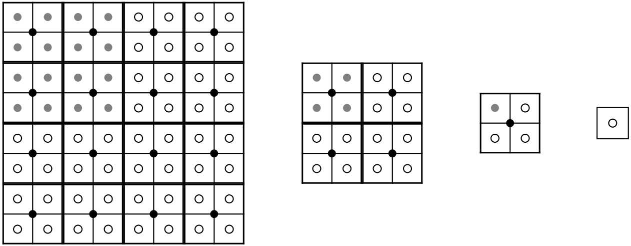
Definition (1/2)



- A $N \times N/q$ M-pyramid is defined by :
  - A reduction window $N \times N$ corresponds to a connected set of pixels used to compute the value of a pixel in image above. The function applied to compute this value is called a reduction function.
    - A pixel is the father of all the pixels belonging to its reduction window.
    - Any pixel within a reduction window is the child of at least one father (see below).
  - The Reduction factor q encodes the ratio between the sizes of two consecutive images. This ratio is fixed along the pyramid.

Definition (2/2)

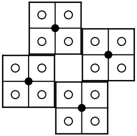
- The Receptive field is defined as the transitive closure of the father/child relationship.

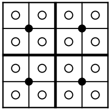


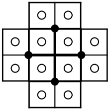
Receptive field

Different types of M-Pyramids

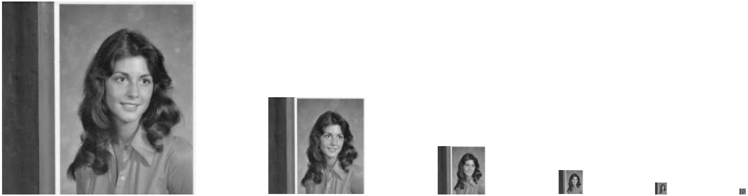
- If $N \times N/q < 1$, the pyramid is named a non-overlapping holed pyramid(a). Some pixels have no fathers (e.g., the center pixel in Fig. (a)).
- If $N \times N/q = 1$, the pyramid is called a non overlapping pyramid without hole(b). Each pixel in the reduction window has exactly one father.
- If $N \times N/q > 1$, the pyramid is named an Overlapping pyramid (c). Each pixel has several potential parents.


a) $2 \times 2/5$

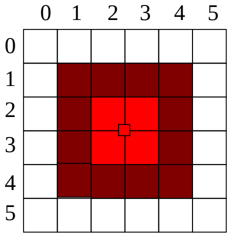

b) $2 \times 2/4$


c) $2 \times 2/2$

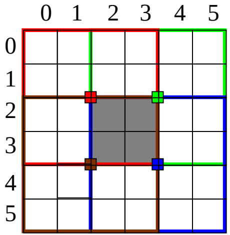
Example : A $2 \times 2/4$ non overlapping pyramid



Overlapping pyramids



- q inner childs,
- $N^2 - q$ outer childs



- $NxN/q > 1$: Each pixel contributes to several father $\Rightarrow$ Each pixel has several potential father

Segmentation algorithm

- Main notations :
  - Legitimate father : Closest father (strongest link w)
  - P' : son of P, P^o : legitimate father of P.
  - Root : Link(P,Legitimate(P))<threshold)
- Algorithm :
  - From Bottom to Top
    - Compute values and

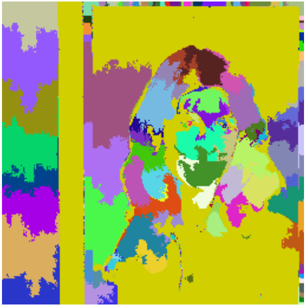
$$v(P) = \frac{\sum_{P'} v(P')a(P')w(P,P')}{\sum_{P'} w(P,P')a(P')} \quad a(P) = \sum_{P'} \frac{a(P')w(P,P'}{\sum_{P'^o} w(P,P'^o)}$$

    - update links untill no change occur.

$$w(P,P^o) = \frac{(v(P) - v(P^o))^{-2}}{\sum_{P^{o\prime}} (v(P^{o\prime}) - v(P^o))^{-2}}$$

  - From Top to Bottom
    - Select roots
    - Link non roots to legitimate fathers

Examples

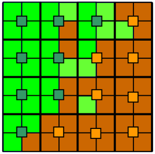


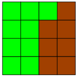
Advantages of Regular pyramids

- reduce the influence of noise
- makes the processes independent of the resolution
- convert global features to local ones
- reduce computational cost
- Analysis at low cost using low resolution images.

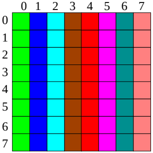
Drawbacks (1/2)

- Shift - Scale-Rotation variant
- Preservation of the connectivity is not guaranteed





- Limited number of regions at a given level

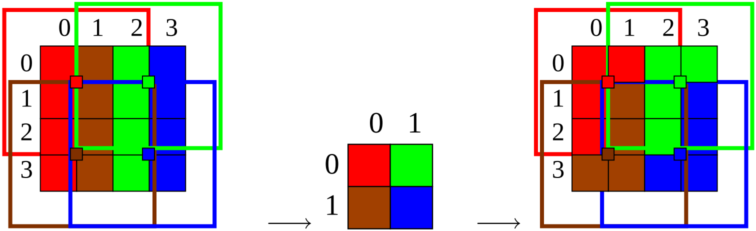


with a $4 \times 4/4$ pyramid :

$$\begin{cases} \text{can be described only at level 3} \\ \text{only 4 pixels left at level 3} \end{cases}$$

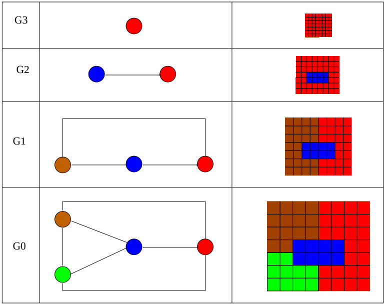
Drawbacks (2/2)

- Difficulties to encode elongated objects



Irregular Pyramids

- Can wee keep all the advantages of regular pyramids while overcoming their drawbacks ?
- Yes, using irregular pyramids :
  - Stack of graphs $(G_0, G_1, \ldots, G_n)$ successively reduced.
  - G_0 : encodes the initial grid or an initial segmentation.



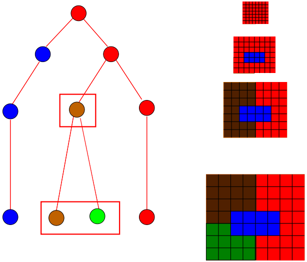
- $G_0, \ldots, G_n$ are only final results.

Reduction window

- $v \in V_i$ comes from the merge of a connected set of vertice in G_{i-1}.

$$RW_i(v) = \{v_1, \ldots, v_n\} \subset V_{i-1}$$

  - $v_j \in RW_i(v)$ is a son of v,
  - v is the father of all $v_j \in RW_j(v)$.

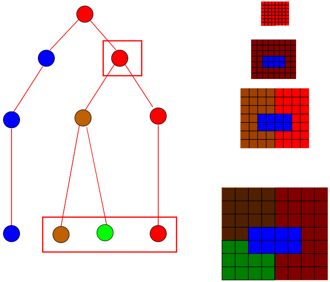


Receptive field

- Receptive field : transitive closure of the father/child relationship.

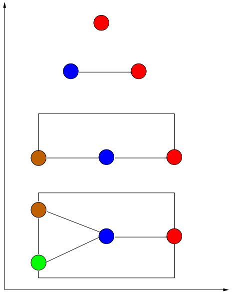
$$RF_i(v) = \bigcup_{v' \in RW_i(v)} RW_{i-1}(v') \subset V_0$$

  - $w \in RF_i(v)$ is a descendant of v,
  - v is an ancestor of w.



Research fields

- Pyramid construction schemes (vertical definition)
  - sequential methods,
  - parallel methods.
    - kernel method [Meer 89],
    - Data driven decimation [Jolion 2001],
    - decimation by maximal matching [Haximusa et al. 2005].
- Encoding of partitions (horizontal definition)
  - simple graphs,
  - dual graphs,
  - combinatorial maps



Construction schemes of the pyramid

- sequential methods :
  - sort the edges of the graphs
  - Union-find
- parallel method :
  - Define parallel merge operations
  - each step builds a new graph G_{i+1} from G_i.
  - $|G_{i+1}|$ is a fixed ratio of $|G_i|$.

$$|G_{i+1}| \approx q|G_i| \text{ with } q < 1 : \text{reduction factor}$$

  - ⚠ the parallelism is a constraint for segmentation algorithms
    - ☹ : “forces” a fixed amount of fusions at each step

$$|G_{i+1}| \approx q|G_i|$$

    - ☺: bounds the number of graphs we have to build/store

$$\mathcal{P} = (G_0 \ldots, G_n) \text{ with } n = log_r(|G_0|)$$

Parallel construction schemes

- A set of independant processes merge vertices in parallel
- Problem : How to insure that : $\frac{V_i}{V_{i-1}} \lessapprox \frac{1}{2}$
  - computational time
  - storage memory.

Kernel methods

- Introduced by Meer in 1989.
- We build a set of surviving vertice which will correspond to the vertice of V_{i+1}.
- V_{i+1} must satisfy two constraints :

  External stability :

  $$\forall v \in V_i - V_{i+1} \ \exists v' \in V_{i+1} : (v, v') \in E_i$$

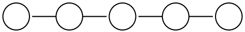
  Each non surviving vertex is adjacent to at least a surviving one

  Internal stability :

  $$\forall (v, v') \in V_{i+1}^2 \ (v, v') \in E_i$$

  Two adjacent vertice cannot both survive

Kernel construction scheme : selection of surviving vertice



- Three variables :

$p_i = true$ if v_i survives

$q_i = true$ if v_i may become a surviving vertex (he is candidate).

x_i value of the vertex (function or random variable)

$$p_i^{(1)} = x_i = max_{j \in V(v_i)}\{x_j\}$$
$$q_i^{(1)} = \bigwedge_{j \in V(v_i)} \overline{p_j}^{(1)}$$
$$p_i^{(k+1)} = p_i^{(k)} \vee (q_i^{(k)} \wedge x_i = max_{j \in V(v_i)}\{q_j^{(k)} x_j\})$$
$$q_i^{(k+1)} = \bigwedge_{j \in V(v_i)} \overline{p_j}^{(k+1)}$$

Kernel construction scheme : selection of surviving vertice

- Three variables :

  $p_i = true$ if v_i survives

  $q_i = true$ if v_i may become a surviving vertex (he is candidate).

  x_i value of the vertex (function or random variable)

$$p_i^{(1)} = x_i = max_{j \in V(v_i)} \{x_j\}$$
$$q_i^{(1)} = \bigwedge_{j \in V(v_i)} \overline{p_j}^{(1)}$$
$$p_i^{(k+1)} = p_i^{(k)} \vee (q_i^{(k)} \wedge x_i = max_{j \in V(v_i)} \{q_j^{(k)} x_j\})$$
$$q_i^{(k+1)} = \bigwedge_{j \in V(v_i)} \overline{p_j}^{(k+1)}$$

Kernel construction scheme : selection of surviving vertice

- Three variables :

  $p_i = true$ if v_i survives

  $q_i = true$ if v_i may become a surviving vertex (he is candidate).

  x_i value of the vertex (function or random variable)

$$p_i^{(1)} = x_i = max_{j \in V(v_i)}\{x_j\}$$
$$q_i^{(1)} = \bigwedge_{j \in V(v_i)} \overline{p_j}^{(1)}$$
$$p_i^{(k+1)} = p_i^{(k)} \vee \left(q_i^{(k)} \wedge \; x_i = max_{j \in V(v_i)}\{q_j^{(k)} x_j\}\right)$$
$$q_i^{(k+1)} = \bigwedge_{j \in V(v_i)} \overline{p_j}^{(k+1)}$$

Kernel construction scheme : selection of surviving vertice

- Three variables :

  $p_i = true$ if v_i survives

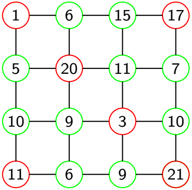
  $q_i = true$ if v_i may become a surviving vertex (he is candidate).

  x_i value of the vertex (function or random variable)

$$
\begin{aligned}
p_i^{(1)} &= x_i = max_{j \in V(v_i)}\{x_j\} \\
q_i^{(1)} &= \bigwedge_{j \in V(v_i)} \overline{p_j}^{(1)}
\end{aligned}
$$

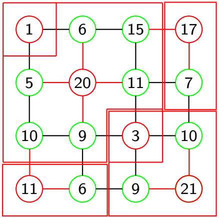
$$
\begin{aligned}
p_i^{(k+1)} &= p_i^{(k)} \vee \left(q_i^{(k)} \wedge \ x_i = max_{j \in V(v_i)}\{q_j^{(k)} x_j\}\right) \\
q_i^{(k+1)} &= \bigwedge_{j \in V(v_i)} \overline{p_j}^{(k+1)}
\end{aligned}
$$

Kernel construction scheme : father/child relationships



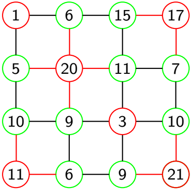
- link each non surviving vertex to one of its surviving neighbour $\Rightarrow$ definition of the edges
- merge non surviving vertice to surviving ones along the selected edges(merge in simple graphs).

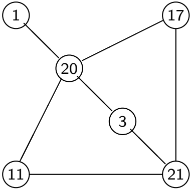
Kernel construction scheme : father/child relationships



- link each non surviving vertex to one of its surviving neighbour ⇒ definition of the edges
- merge non surviving vertice to surviving ones along the selected edges(merge in simple graphs).

Kernel construction scheme : father/child relationships





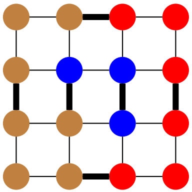
- link each non surviving vertex to one of its surviving neighbour $\Rightarrow$ definition of the edges
- merge non surviving vertice to surviving ones along the selected edges(merge in simple graphs).

Maximal matching : Motivations

- Method introduced by Haximusa & Kropatsch $\approx$ 2005
- within the kernel construction scheme the probability that a vertex survives decreases with its degree.
- The mean degree of vertices increases within the pyramid.
- $\Rightarrow$ The ratio $\frac{V_i}{V_{i-1}}$ computed by the kernel method decreases according to the level
  - Increases the computational time, even on parallel processors.
  - Useless graph storage.

Maximal matching

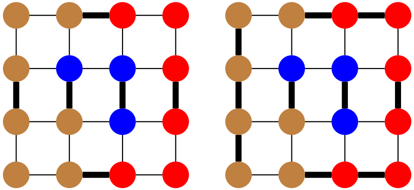
- Define a maximal matching C(kernel of $G' = (E, E')$)
  - $(e, e') \in E'$ iff e and e' are incident to a same vertex.
- Complete the matching C to C^+
- Remove edges from C^+ in order to obtain trees of depth 1.
- Merge vertice adjacent along selected edges.



- A set $C \subset E$ is said to be a matching of $G = (V, E)$ if none of the edges of C are adjacent to a same vertex.
- A matching is said to be maximal if the addition of any edge breaks the matching property.
- A matching is said to be maximum if no larger matching may be found.

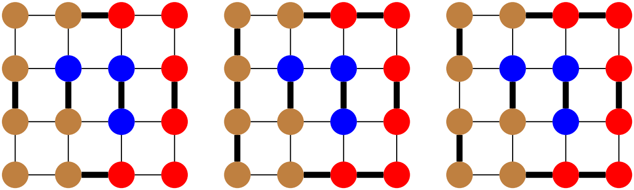
Maximal matching

- Complete the matching C to C^+
- Remove edges from C^+ in order to obtain trees of depth 1.
- Merge vertice adjacent along selected edges.



Maximal matching

- Complete the matching C to C^+
- Remove edges from C^+ in order to obtain trees of depth 1.
- Merge vertice adjacent along selected edges.



Data driven decimation

- perform on iteration of the kernel computation,
- attach each non surviving vertex to a surviving one
- merge vertices
- continue on the reduced graph
- Method introduced by Jolion $\approx$ 2001

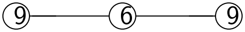
Data driven decimation

- perform on iteration of the kernel computation,
- attach each non surviving vertex to a surviving one
- merge vertices
- continue on the reduced graph
- Method introduced by Jolion $\approx$ 2001

Data driven decimation

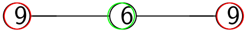
- perform on iteration of the kernel computation,
- attach each non surviving vertex to a surviving one
- merge vertices
- continue on the reduced graph
- Method introduced by Jolion $\approx$ 2001

Data driven decimation



- perform on iteration of the kernel computation,
- attach each non surviving vertex to a surviving one
- merge vertices
- continue on the reduced graph
- Method introduced by Jolion $\approx$ 2001

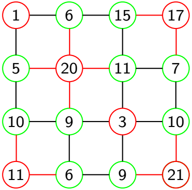
Data driven decimation



- perform on iteration of the kernel computation,
- attach each non surviving vertex to a surviving one
- merge vertices
- continue on the reduced graph
- Method introduced by Jolion $\approx$ 2001

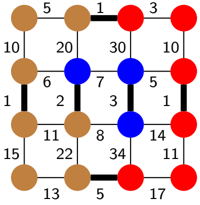
Data driven decimation : conclusion

- only one step of the kernel computation is performed
  - "Corresponds" to a model of the behavior of our brain,
  - allows to avoid (in some cases) wrong merge operations.



Data driven decimation and maximal matching

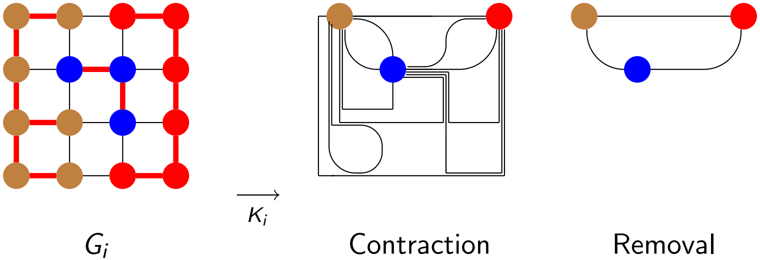
- Method introduced by Pruvot & Brun
- The maximal matching is defined as a MIS on the graph $G = (E, E')$.
  1. Value each edge as a merging cost,
  2. Perform only one iteration of the maximal matching algorithm
  3. One edge is selected if it is locally minimal (the two regions like each other more than any of their neighbour).



Simple graph pyramids

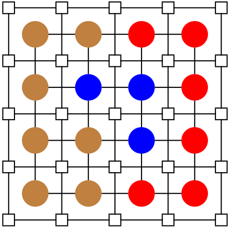
$G_i \rightarrow G_{i+1}$ by a merge operation between vertice

- Construction scheme :
  1. Define $K_i \subset E_i$ (c.f. previous slides)
  2. Contract K_i
  3. Remove any loops and double edges



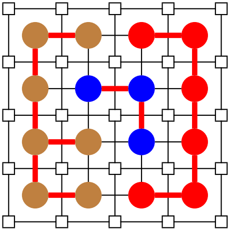
Construction scheme of dual graph pyramids

- $G_i = G_0$
- Define a set K_i of edges to be contracted



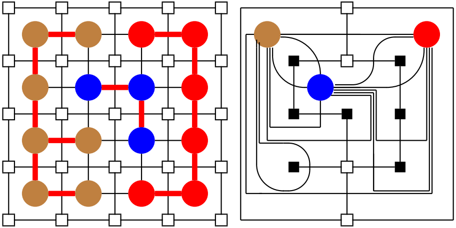
Construction scheme of dual graph pyramids

- $G_i = G_0$
- Define a set K_i of edges to be contracted
  - K_i must be a forest of G_i (we do not contract loops) called a contraction kernel



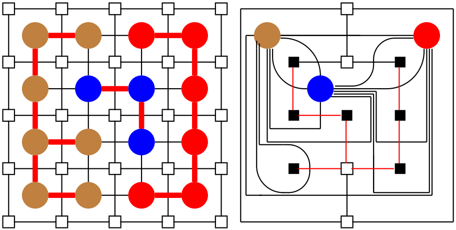
Construction scheme of dual graph pyramids

- $G_i = G_0$
- Define a set K_i of edges to be contracted
- Contract K_i within $G_i \rightarrow G_{i+1}$,
- Define a set K_{i+1} of edges to remove
- Contract K_{i+1} within $\overline{G_{i+1}} \rightarrow G_{i+2}$.



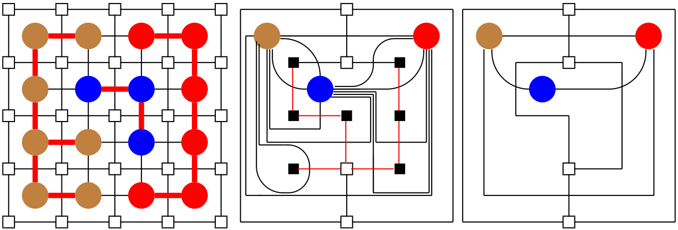
Construction scheme of dual graph pyramids

- $G_i = G_0$
- Define a set K_i of edges to be contracted
- Contract K_i within $G_i \rightarrow G_{i+1}$,
- Define a set K_{i+1} of edges to remove
  - K_{i+1} is a forest of $\overline{G_{i+1}}$ called a Removal kernel
  - $e \in K_{i+1} \Rightarrow e$ is incident to $f \in \overline{V_{i+1}}$, $d^\circ(f) \leq 2$.
- Contract K_{i+1} within $\overline{G_{i+1}} \rightarrow G_{i+2}$.



Construction scheme of dual graph pyramids

- $G_i = G_0$
- Define a set K_i of edges to be contracted
- Contract K_i within $G_i \rightarrow G_{i+1}$,
- Define a set K_{i+1} of edges to remove
- Contract K_{i+1} within $\overline{G_{i+1}} \rightarrow G_{i+2}$.



Comparison of dual and simple graph pyramidal construction schemes

- Construction :

	Simple Pyr.	Dual Graph Pyr.
Etap 1	Define K	Define K
Etap 2	Merge according to K	Contract K within G
Etap 3		Define $\overline{K}$
Etap 4		Contract $\overline{K}$ within $\overline{G}$

- Information associated to edges :
  - Simple graphs : adjacency between regions.
  - Dual graphs : boundary information
- Reduction window, receptive fields : same definitions in both cases.

Comparison of dual and simple graph pyramidal construction schemes

- Construction :

	Simple Pyr.	Dual Graph Pyr.
Etap 1	Define K	Define K
Etap 2	Merge according to K	Contract K within G
Etap 3		Define $\overline{K}$
Etap 4		Contract $\overline{K}$ within $\overline{G}$

- Information associated to edges :
  - Simple graphs : adjacency between regions.
  - Dual graphs : boundary information
- Reduction window, receptive fields : same definitions in both cases.

Comparison of dual and simple graph pyramidal construction schemes

- Construction :

	Simple Pyr.	Dual Graph Pyr.
Etap 1	Define K	Define K
Etap 2	Merge according to K	Contract K within G
Etap 3		Define $\overline{K}$
Etap 4		Contract $\overline{K}$ within $\overline{G}$

- Information associated to edges :
  - Simple graphs : adjacency between regions.
  - Dual graphs : boundary information
- Reduction window, receptive fields : same definitions in both cases.

Combinatorial Pyramids : Construction

- Same contruction scheme than for the dual graph pyramids
  1. Definition of a contraction kernel K of G,
  2. Definition of two removal kernels $\overline{K_1}$ and $\overline{K_2}$ of $\overline{G}$ removing respectively :
     - the empty loops ($|\varphi^*(b)| = 1$) and
     - the double edges ($|\varphi^*(b)| = 2$).
- $P = (G_0, \ldots, G_n)$ G_i is deduced from G_{i-1} by a contraction or a removal kernel

$$\mathcal{D}_n \subset \mathcal{D}_{n-1} \subset \cdots \subset \mathcal{D}_0$$

Combinatorial Pyramids and graphs

- One single map $G_i = (\mathcal{D}_i, \sigma_i, \alpha_i)$ instead of two graphs $(G_i, \overline{G_i})$ at each level.
- Reduction window problem :
  - ⚠ a vertex of G_i is defined by a cycle $\sigma_i^*(b), b \in \mathcal{D}_i$
  - $\Rightarrow$ No explicit encoding of vertice.
- Two solutions :
  - create a labeling (explicit encoding) of vertice
  - Speak map language

Combinatorial Pyramids and graphs

- One single map $G_i = (\mathcal{D}_i, \sigma_i, \alpha_i)$ instead of two graphs $(G_i, \overline{G_i})$ at each level.
- Reduction window problem :

$$RF_i(v) = \{v_1, \ldots, v_n\}$$

  - ⚠ a vertex of G_i is defined by a cycle $\sigma_i^*(b), b \in \mathcal{D}_i$
  - $\Rightarrow$ No explicit encoding of vertice.
- Two solutions :
  - create a labeling (explicit encoding) of vertice
  - Speak map language

Combinatorial Pyramids and graphs

- One single map $G_i = (\mathcal{D}_i, \sigma_i, \alpha_i)$ instead of two graphs $(G_i, \overline{G_i})$ at each level.
- Reduction window problem :
  - ⚠ a vertex of G_i is defined by a cycle $\sigma_i^*(b), b \in \mathcal{D}_i$
  - $\Rightarrow$ No explicit encoding of vertice.
- Two solutions :
  - create a labeling (explicit encoding) of vertice ☹
  - Speak map language ☺

Combinatorial Pyramids and graphs

- One single map $G_i = (\mathcal{D}_i, \sigma_i, \alpha_i)$ instead of two graphs $(G_i, \overline{G_i})$ at each level.
- Reduction window problem :
  - ⚠ a vertex of G_i is defined by a cycle $\sigma_i^*(b), b \in \mathcal{D}_i$
  - $\Rightarrow$ No explicit encoding of vertice.
- Two solutions :
  - create a labeling (explicit encoding) of vertice ☹
  - Speak map language ☺

Combinatorial Pyramids and graphs

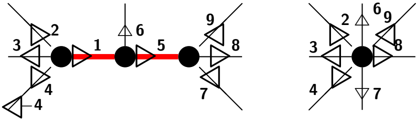
- One single map $G_i = (\mathcal{D}_i, \sigma_i, \alpha_i)$ instead of two graphs $(G_i, \overline{G_i})$ at each level.
- Reduction window problem :
  - ⚠ a vertex of G_i is defined by a cycle $\sigma_i^*(b), b \in \mathcal{D}_i$
  - $\Rightarrow$ No explicit encoding of vertice.
- Two solutions :
  - create a labeling (explicit encoding) of vertice ☹
  - Speak map language ☺

Connecting walks :

- Let $b \in \mathcal{D}_i$, $CW_i(b)$: sequence of darts to traversee within G_{i-1} in order to connect b to
  - $\varphi_i(b)$ if K_{i-1} is a contraction kernel
  - $\sigma_i(b)$ si K_{i-1} is a removal kernel.

Connecting walks :

- If K_{i-1} is a contraction kernel :



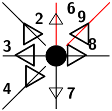
$$CW_i(b) = b\varphi_{i-1}(b)\dots\varphi_{i-1}^{n-1}(b);$$

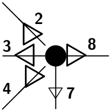
with $n = Min\{k \in \mathbb{N}^* \mid \varphi_{i-1}^k(b) \in \mathcal{D}_i\}$,
$CW_i(-4) = -4.1.5.$

- If K_{i-1} is a removal kernel

Connecting walks :

- If K_{i-1} is a removal kernel





$$CW_i(b) = b.\sigma_{i-1}(b) \ldots \sigma_{i-1}^{n-1}(b)$$

with $n = Min\{k \in \mathbb{N}^* \mid \sigma_{i-1}^{k}(b) \in \mathcal{D}_i\}$.
$CW_i(8) = 8.9.6$

Connecting walks

- In short :

$$CW_i(b) = b.b_1.\dots.b_p$$

- $\{b_1, \dots, b_p\} \subset \mathcal{D}_{i-1}$

$$\begin{array}{lcll} \varphi_i(b) & = & \varphi_{i-1}(b_p) & \text{If } K_{i-1} \text{ is a contraction kernel} \\ \sigma_i(b) & = & \sigma_{i-1}(b_p) & \text{If } K_{i-1} \text{ is a removal kernel} \end{array}$$

- the connecting walks allows to compute G_i from G_{i-1} and K_i

Connecting dart sequences

- Receptive field : transitive closure of reduction windows.

$$CR_i(v) = \bigcup_{v' \in FR_i(v)} CR_{i-1}(v') \subset V_0$$

- Connecting dart sequences : closure (concatenation) of connecting walks

$$SC_i(b) = SC_{i-1}(b_1) \ldots . SC_{i-1}(b_p) \subset \mathcal{D}_0$$

with $CW_i(b) = b_1 \ldots . b_p$.

- nb : This last formula is only valid when two consecutive kernels (K_{i-1}, K_i have a same type (both contraction or removal) kernels.

Connecting dart sequences

- Let $G_0 = (\mathcal{D}_0, \sigma_0, \alpha_0)$.

$$\forall b \in \mathcal{D}_0 \; SC_0(b) = b$$

- for all i in $\{1, \dots, n\}$

$$\forall b \in \mathcal{D}_i \; SC_i(b) = b_1.SC^*_{i-1}(\alpha_{i-1}(b_1)) \dots .b_p.SC^*_{i-1}(\alpha_{i-1}(b_p))$$

with

  - $CW_i(b) = b_1 \dots .b_p$,
  - $SC^*_{i-1}(b_j)$: $SC_{i-1}(b_j)$ minus its first dart (b_j).

- $SC_i(b)$ is defined in G_0.

$$\forall b \in \mathcal{D}_1 \; SC_1(b) = CW_1(b)$$

Connecting dart sequence : Properties

$$CW_i^*(b) \subset K_i \text{ and } SC_i^*(b) \subset \bigsqcup_{j=0}^{i-1} K_j$$

- $b \in \mathcal{D}_i$, $SC_i(b) = b.b_1 \ldots, b_p$, $p > 1$
  - If K_i is a contraction kernel :

$$\varphi_i(b) = \begin{cases} \varphi_0(b_p) & \text{If } b_p \text{ is contracted} \\ \sigma_0(b_p) & \text{Si } b_p \text{ is removed.} \end{cases}$$

  - If K_i is a removal kernel :

$$\sigma_i(b) = \begin{cases} \varphi_0(b_p) & \text{If } b_p \text{ is contracted} \\ \sigma_0(b_p) & \text{If } b_p \text{ is removed.} \end{cases}$$

- May be extended to any $j \leq i \rightarrow (G_0, \ldots, G_n)$

Connecting dart sequences : traversal

- Let $SC_i(b) = b.b_1, \ldots, b_p,\ p > 1$:

$$b_1 = \begin{cases} \varphi(b) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma(b) & \text{If } K_i \text{ is a removal kernel} \end{cases}$$

et

$$\forall j \in \{2, \ldots, p\} \quad b_j = \begin{cases} \varphi(b_{j-1}) & \text{Si } b_{j-1} \text{ is contracted} \\ \sigma(b_{j-1}) & \text{Si } b_{j-1} \text{ is supprimed.} \end{cases}$$

- We need to know :
  - The type of K_i,
  - The operation aplied to each dart.

Implicit encoding

- Let the two functions :
  - **state**

$$\left(\begin{array}{ccl} \{1,\dots,n\} & \rightarrow & \{0,1,2\} \\ i & \mapsto & \left\{\begin{array}{ll} 0 & \text{If } K_i \text{ cont. kernel} \\ 1 & \text{If } K_i \text{ rem. kernel (empty selft loops)}; \\ 2 & \text{If } K_i \text{ rem. kernel (double edges)} \end{array}\right. \end{array}\right.$$

    In practice $\textbf{state}(i) = i \ mod \ 3$
  - **level** :

$$\left(\begin{array}{ccl} \mathcal{D}_0 & \rightarrow & \{1,\dots,n+1\} \\ b & \mapsto & \max\{i \in \{1,\dots,n+1\} \mid b \in \mathcal{D}_{i-1}\}. \end{array}\right.$$

- In terms of encoding :
  - **state** : arrray of bits of size n.
  - **level** :array of integers of size $|\mathcal{D}_0|$.

Implicit encoding : Definition

- Explicit encoding $P = (G_0, \ldots, G_n)$
- Implicit encoding : $P = (G_0, \textbf{state}, \textbf{level})$.
  - Any may G_i may be retreived from the implicit encoding
  - No more restriction on the number fo levels
  - Maximal compression :
    - G_0 : 4 connected grid $\rightarrow$ implicitly encoded
    - : **state**(i) : *i mod* 3
    - Encoding : $P = (\textbf{level})$
  - For practical reasons :
    - $P = (G_0, G_n, \textbf{level})$ or
    - $P = (G_n, \textbf{level})$.

Implicit encoding

- Trversal of $SC_i(b) = b.b_1 \ldots .b_p$ using **state**and **level** :

$$b_1 = \begin{cases} \varphi_0(b) & \text{If } \textbf{state}(\text{i}) = \text{contracted} \\ \sigma_0(b) & \text{If } \textbf{state}(\text{i}) = \text{removed} \end{cases}$$

et

$$b_j = \begin{cases} \varphi_0(b_{j-1}) & \text{If } \textbf{state}(\textbf{level}(b_{j-1})) = \text{Contracted} \\ \sigma_0(b_{j-1}) & \text{If } \textbf{state}(\textbf{level}(b_{j-1})) = \text{Removed} \end{cases}$$

for $j \in \{2, \ldots, p\}$

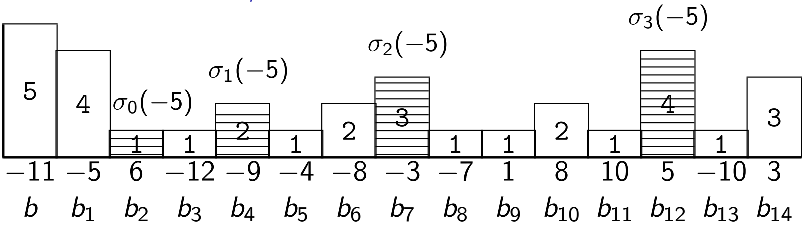
Conversion Implicit/Explicit

- Solution 1 : Compute successively each map $G_1, \ldots, G_n$ ☹.
- Solution 2 : study the structure of the connecting dart sequences

$$SC_i(b) = b_1.SC^*_{i-1}(\alpha_{i-1}(b_1)) \ldots .b_p.SC^*_{i-1}(\alpha_{i-1}(b_p))$$

  - Computation of all maps $(G_0, \ldots, G_i)$ in one traversal of the connecting dart sequences of level i.

Conversion Implicit/Explicit



- If $\mathcal{D}_n = \{b, \alpha_n(b)\}$, the whole pyramid is conmupted by one traversal of $SC_n(b)$ et $SC_n(\alpha_n(b))$.

$$\forall i \in \{0, \dots, n\} \bigsqcup_{b \in \mathcal{D}_i} SC_i(b) = \mathcal{D}_0$$

Vertex receptive fields

- Can we obtain intermediate results between the darts and the whole map ?
- Let $\sigma_i^*(b) = (b_1, \ldots, b_p)$. We want to connect b_j to $\sigma_i(b_j) = b_{j+1}$ in G_0.

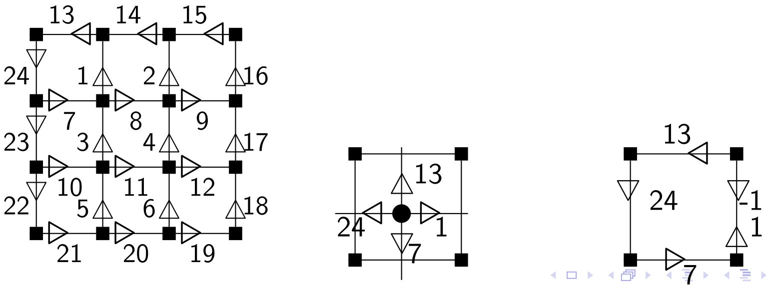
$$RF_i(b) = \begin{cases} SC_i(b) & \text{If } K_i \text{ is a removal kernel} \\ b.SC_i^*(\alpha(b)) & \text{If } K_i \text{ is a contraction kernel.} \end{cases}$$

- Receptive field of $\sigma_i^*(b)$:

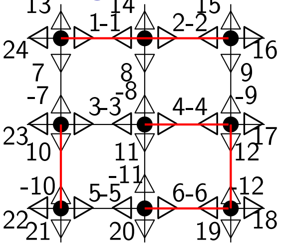
$$R_{\sigma_i^*(b)} = \bigodot_{j=1}^{p} RF_i(b_j)$$

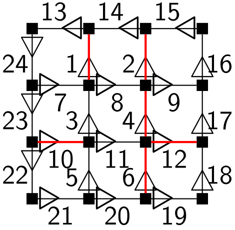
Embedding

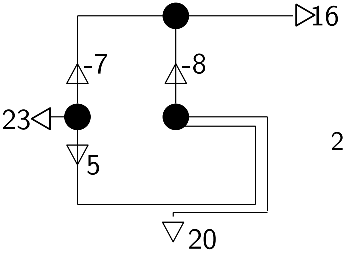
- If G_0 encode the 4 connected grid
  - $\sigma_0^*(b)$ corresponds to a pixel,
  - $\alpha_0^*(b)$ corresponds to a lignel,
    - b corresponds to an oriented lignel.
  - $\varphi_0^*(b)$ corresponds to a pointel.

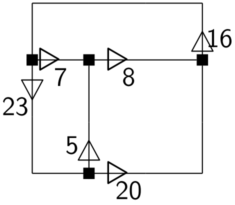


Embedding of receptive fields

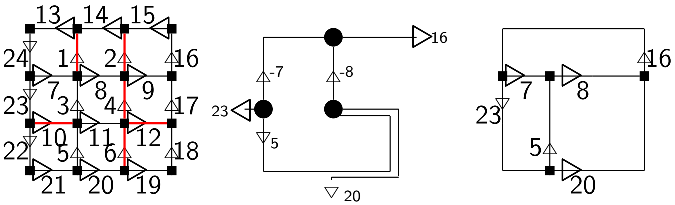








Embedding of receptive fields



- $\sigma_i^*(16) = (16, 7, 8)$

$$R_{\sigma_i^*(16)} = 16.15.-2.14.-1.13.24.7.1.8.2.9$$

Traversal of borders

- Traversal of darts encoding lignel borders :

$$\partial R_{\sigma_i^*(b)} = b_1, \ldots, b_p \text{ avec } \left\{ \begin{array}{lcl} b_1 & = & b \\ b_j & = & \varphi_0^{n_j}(\alpha_0(b_j)) \end{array} \right.$$

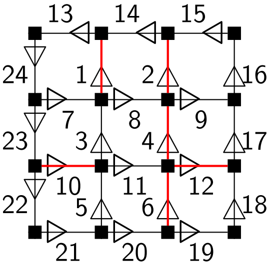
  where $n_j = Min\{p \in \mathbb{N}^* \,|\varphi_0^p(\alpha_0(b_j)) \in \mathcal{D}_i \text{ or double edge}\}$.
- Rem : b double edge $\Leftrightarrow$ **level**$(b) mod 3 = 2$(cont., rem. empty self-loops, rem. double edges).
- The border is included within the region $\partial R_{\sigma_i^*(b)} \subset R_{\sigma_i^*(b)}$

Traversal of borders

$$\partial R_{\sigma_i^*(b)} = b_1, \ldots, b_p \text{ avec } \left\{ \begin{array}{lcl} b_1 & = & b \\ b_j & = & \varphi_0^{n_j}(\alpha_0(b_j)) \end{array} \right.$$

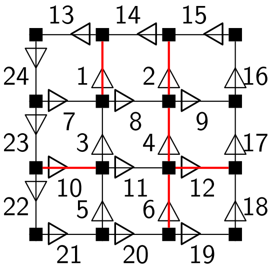
$$R_{\sigma_i^*(16)} = 16.15.-2.14.-1.13.24.7.1.8.2.9$$

$$\partial R_{\sigma_i^*(16)} = 16.15.14.13.24.7.8.9$$



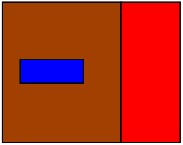
Traversal of borders

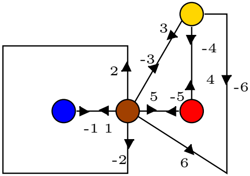
- If we modify the operation of self loops removal :
  - $SC_i(16) = 16.15.{-2}.14.{-1}.13.24$ $\rightarrow \partial SC_i(16) = 16.15.14.13.24$
  - $SC_i(7) = 7.1 \rightarrow \partial SC_i(7) = 7$
  - $SC_i(8) = 8.2.9 \rightarrow \partial SC_i(8) = 8.9$



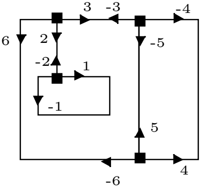
Inside relationships

- Inside relationships are characterized by the loops





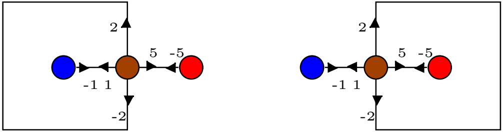
$G = (\mathcal{D}, \sigma, \alpha)$



$\overline{G} = (\mathcal{D}, \varphi, \alpha)$

Inside relationships

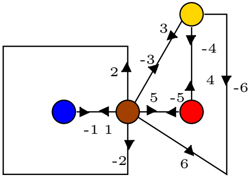
- But the location of loops is ambigous.



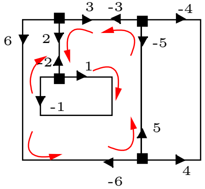
$$\sigma = (2, 1, -2, 5)(-1)(-5)$$

Inside relationships

- Solution : Use the orientation



$G = (\mathcal{D}, \sigma, \alpha)$



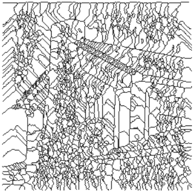
$\overline{G} = (\mathcal{D}, \varphi, \alpha)$

Complexities

- Coputation of one map : $\mathcal{O}$ (size of the borders),
- Computation of all maps : $\mathcal{O}(\mathcal{D}_0) \approx \mathcal{O}(|I|)$
- Traversal of one border : $\mathcal{O}$ (size (in lignels) of the border)
- Inside relationships $\mathcal{O}(|\sigma_i^*(b)|)$.

Application 1 : hierarchical Watershed

Image

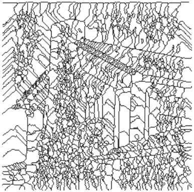


LPE

- hierarchical Watershed :
  1. Compute the watershed
  2. valuate the importance of each contour
     2.1 minimal value of the gradient along the contour...
  3. sort the contours :
     - remove the less significative ones.
     - examin the merge of regions according to the importance of the contours.

Application 1 : hierarchical Watershed

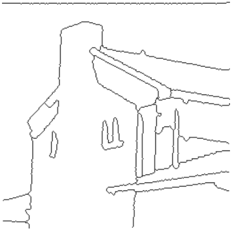
Image



LPE

Dynamics

- Improvements :
  1. Taking into account the evolution of the partition
     - implicit encoding
  2. robust valuation of the contours
     - geometric embedding of the darts



Segmentation

Application 2 : Inside relationships