Partitions & non hierarchical models

Luc Brun

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

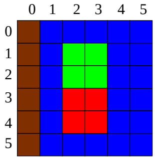
Plan

Partition

- An image's partition is defined as a set of regions that collectively cover the entire image and whose intersection of any couple of region is empty.

$$\mathcal{P} = \{R_1, \ldots, R_n\}$$

$$\forall i \neq j \; R_i \cap R_j = \emptyset$$
$$P = \bigsqcup_{i=1}^{n} R_i,$$

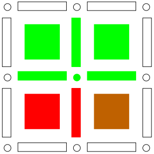


Partitions and Kovalevsky's Topology

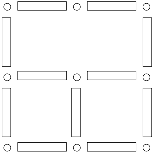
- Maximum label rule : Let l be a real function defined over all cells of maximal dimension.

$$\forall e \in C \; dim(e) < dim_{max} \; l(e) = \max_{\substack{e' \in St(e,C), \\ dim(e') = dim_{max}}} l(e')$$

- Example $\square >$ ■>■>■



$\{R_1, \ldots, R_n\}$



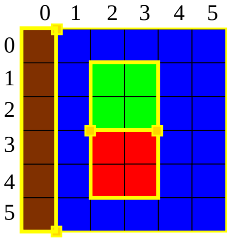
$\bigcup_{i=1}^{n} \partial R_i$

Structuring boundaries

- Node: pointel whose star contains at least 3 different labels.

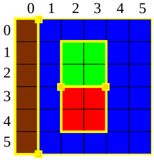
$$\forall e \in C,\ dim(e) = 0, \text{est un noeud} \Leftrightarrow |l(St(e, C))| \geq 3$$

- Segment : maximal sequence of bordering pointels/lignels between two nodes.

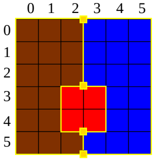


Adjacency

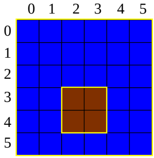
- Two regions are said adjacent if they share at some boundary elements of dimension 1 (i.e. at least one segment).
- Regions ■ and ■ are adjacent in the three cases bellow:



1 segment



2 segment

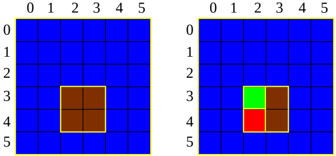


inside

- The merging of two adjacent connected set of pixels (regions) produces a connected set of pixel..

Connected component

- A connected component of a partition is a connected set of regions inside one region of the partition.
- Examples :



Segmentation and Partitions

- Segmentation: Aims to define a partition $\mathcal{P} = \{R_1, \dots, R_n\}$ which satisfy some criteria:
  - Homogeneity of each region:
  $$\forall i \in \{1, \dots, n\},\ P(R_i) = vrai,$$
  - Minimisation of an energy:
  $$\mathcal{P} = argmin_{P \in \mathbb{P}} E(P)$$
  - Binary partition: Find S which minimises:
  $$h(S) = \frac{\int_{\partial S} w(\lambda) d\lambda}{min\left(\int_S w'(x,y)dxdy, \int_{\Omega - S} w'(x,y)dxdy\right)}$$
  If $w = w' = 1$ et $\Omega = \mathbb{R}^2$, it is equivalent to find the form whose perimeter is minimal and which surrounds a maximal volume (i.e. the disc). Isoperimetric problem.
  - ...

Segmentation according to Horowitz

Definition

$\{R_1, \ldots, R_n\}$ is a segmentation of I according to an homogeneity criterion P iff:

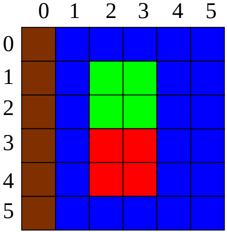
1. $\{R_1, \ldots, R_n\}$ defines a partition of I: $I = \bigsqcup_{i=1}^{n} R_i$
2. in sets connected (regions) $\forall i \in \{1, \ldots, n\}$ R_i is connected
3. and homogeneous $\forall i \in \{1, \ldots, n\}$ $P(R_i) = true$
4. and which is maximal for these properties

$$\forall i \in \{1, \ldots, n\}^2 \begin{pmatrix} i \neq j \\ R_i \text{ adj } R_j \end{pmatrix} P(R_i \cup R_j) = false$$

Segmentation and partitions

- Segmentation algorithms need:
  1. to extract information from partitions and to
  2. modify them.
- "Geometrical" information: Any information on one region that may be deduced solely from the region (without using information from the partition).
  1. Set of pixels of one region (mean, variance, shape,...),
  2. ownership of a point,
  3. Border...
- "Topological" information: Any information that takes sense only when considering partitions.
  1. Border between two regions,
  2. Set of regions adjecent to a region,
  3. Set of connected components inside a region,
  4. region surrounding a connected component...

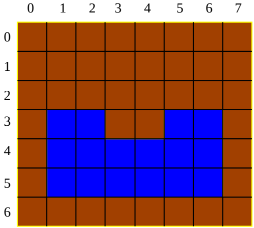
Array of labels

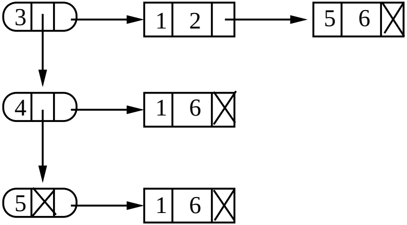


0	1	1	1	1	1
0	1	2	2	1	1
0	1	2	2	1	1
0	1	3	3	1	1
0	1	3	3	1	1
0	1	1	1	1	1

- Advantages:
  - Extremely simple,
  - Access to most of geometrical information
- Drawbacks:
  - No straightforward access to information related to boundaries
  - Not compact
- Remark : Levels sets : $sgn(\phi(x))$: 2 labels.

Run Length Encoding

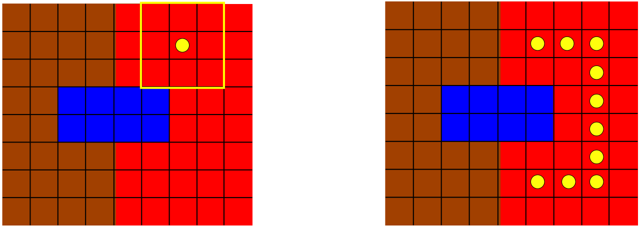




- Advantages:
  - Compact data structure,
  - Straightforward retrieval of the set of pixels,
- Drawbacks
  - No bordering information,
  - Not so easy to update.

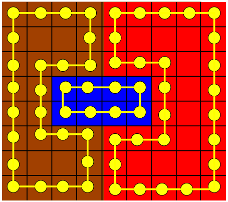
Medial axis transform

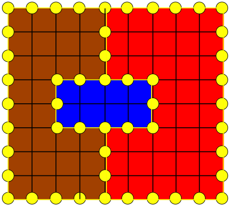
- BB-MAT (or DB-MAT) largest Block (or Disc) inside a region.



- Advantages:
  - Compact representation,
  - Medial Axis : Homotopic transformation from a 2D set to a 1D one which preserves information about the shape of the region ⇒ Shape Recognition.
- Drawbacks
  - Medial axis transform not continuous ⇒ Sensible to small perturbations of the shape.

Borders

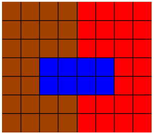




- Advantages :
  - Information about borders,
- Drawbacks (∂ pixel):
  - The location of the border is ambiguous.
  - Redundant information.

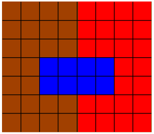
Region Adjacency Graph

- $G = (V, E)$: A simple graph
  - Without loops, 
  - Without double edges, 
  - V set of vertice. One vertex per region
  - E set of edges. One edge per adjacency relationship between regions.



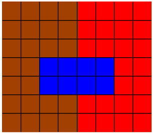
Region Adjacency Graph

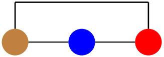
- $G = (V, E)$: A simple graph
  - V set of vertice. One vertex per region
  - E set of edges. One edge per adjacency relationship between regions.



Region Adjacency Graph

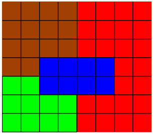
- $G = (V, E)$: A simple graph
  - V set of vertice. One vertex per region
  - E set of edges. One edge per adjacency relationship between regions.

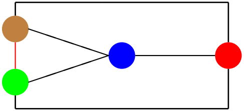




Merge of vertices

- Select one edge encoding the adjacency between both region

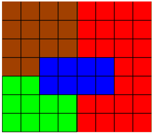


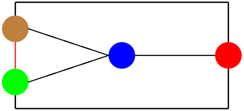


  -
  -
  -

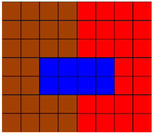
Merge of vertices

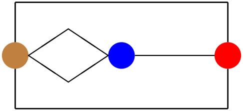
- Select one edge encoding the adjacency between both region





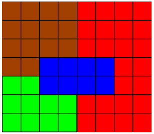
  - Contract the edge (Identify both vertices, remove the edge)
  -
  -

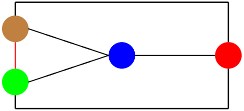




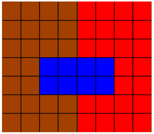
Merge of vertices

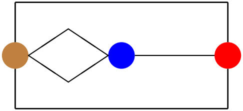
- Select one edge encoding the adjacency between both region





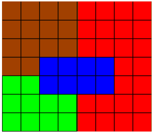
  - Contract the edge (Identify both vertices, remove the edge)
  - Remove any loops that may have appeared
  -

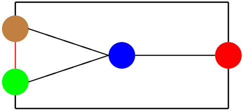




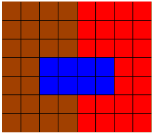
Merge of vertices

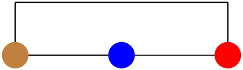
- Select one edge encoding the adjacency between both region





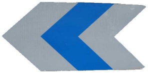
  - Contract the edge (Identify both vertices, remove the edge)
  - Remove any loops that may have appeared
  - Remove any double edge that may have appeared

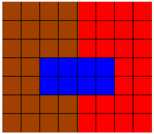


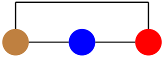


Limits of simple graphs

- Soit $G = (V, E)$,
  - $e = (u, v) \in E \Rightarrow R_u$ and R_v have at least one common border
  - $\Leftrightarrow R_u$ and R_v may be merged.



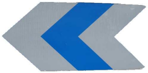


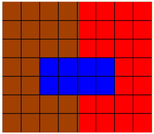


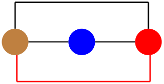
- Not so easy to use for boundary based criteria or criteria using boundary information (amoung other features) ☹.
- Solution : Adds edges. But. . .

Limits of simple graphs

- Soit $G = (V, E)$,
  - $e = (u, v) \in E \Rightarrow R_u$ and R_v have at least one common border
  - $\Leftrightarrow R_u$ and R_v may be merged.

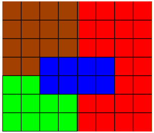


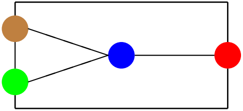




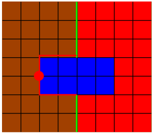
- Not so easy to use for boundary based criteria or criteria using boundary information (amoung other features) ☹.
- Solution : Adds edges. But...

Limits of simple graphs: Illustration



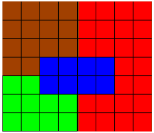


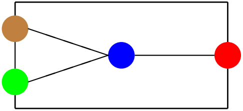
- Identify two adjacent vertice, remove the edge,
- Remove loops,
- Remove double edges.



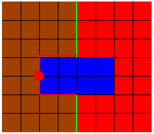
- Redundant double edges “ surround” nothing.

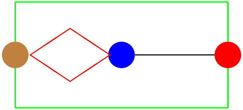
Limits of simple graphs: Illustration





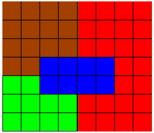
- Identify two adjacent vertice, remove the edge,
- Remove loops,
- Remove double edges.

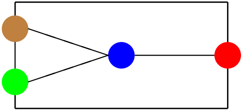




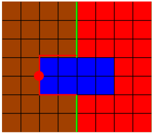
- Redundant double edges " surround" nothing.

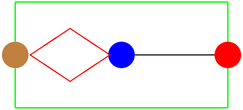
Limits of simple graphs: Illustration





- Identify two adjacent vertice, remove the edge,
- Remove loops,
- Remove double edges.

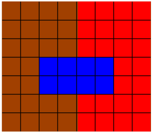


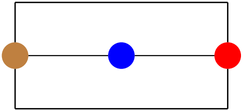


- Redundant double edges “ surround” nothing.

Dual Graphs: Definition

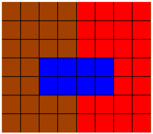
- Dual Graph model: $(G, \overline{G})$
- $G = (V, E)$ non simple,
  - ◯ encode image background
- $\overline{G} = (\overline{V}, \overline{E})$
  - $\overline{V}$: one vertex of $\overline{G}$ per face of G.
  - $\overline{E}$: Each $\overline{e} \in \overline{E}$ cuts one and only one edge of E..

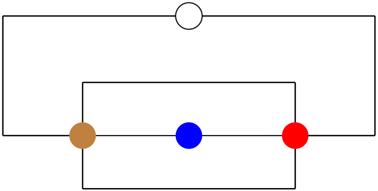




Dual Graphs: Definition

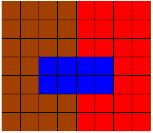
- Dual Graph model: $(G, \overline{G})$
- $G = (V, E)$ non simple,
  - ◯ encode image background
- $\overline{G} = (\overline{V}, \overline{E})$
  - $\overline{V}$: one vertex of $\overline{G}$ per face of G.
  - $\overline{E}$: Each $\overline{e} \in \overline{E}$ cuts one and only one edge of E..

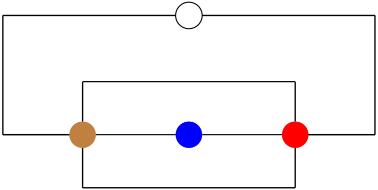




Dual Graphs: Definition

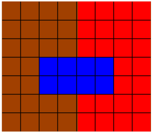
- Dual Graph model: $(G, \overline{G})$
- $G = (V, E)$ non simple,
  - ◯ encode image background
- $\overline{G} = (\overline{V}, \overline{E})$
  - $\overline{V}$: one vertex of $\overline{G}$ per face of G.
  - $\overline{E}$: Each $\overline{e} \in \overline{E}$ cuts one and only one edge of E..

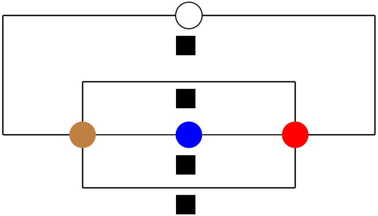




Dual Graphs: Definition

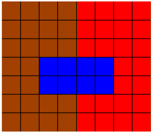
- Dual Graph model: $(G, \overline{G})$
- $G = (V, E)$ non simple,
  - ◯ encode image background
- $\overline{G} = (\overline{V}, \overline{E})$
  - $\overline{V}$: one vertex of $\overline{G}$ per face of G.
  - $\overline{E}$: Each $\overline{e} \in \overline{E}$ cuts one and only one edge of E..

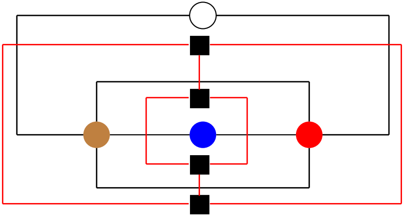




Dual Graphs: Definition

- Dual Graph model: $(G, \overline{G})$
- $G = (V, E)$ non simple,
  - ◯ encode image background
- $\overline{G} = (\overline{V}, \overline{E})$
  - $\overline{V}$: one vertex of $\overline{G}$ per face of G.
  - $\overline{E}$: Each $\overline{e} \in \overline{E}$ cuts one and only one edge of E..



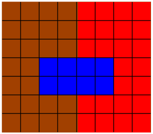


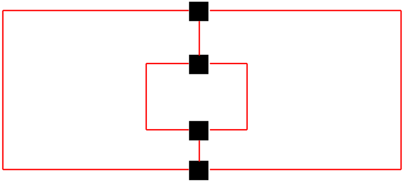
Dual graphs: properties

- The dual operator is an involution :$\overline{\overline{G}} = G$
- We have a $1-1$ correspondance between the edge of G and the ones of $\overline{G}$
- A loop of G is a bridge of $\overline{G}$ and vice versa.
- Any contraction in G implies a removal in $\overline{G}$
- Any removal in G implies a contraction in $\overline{G}$

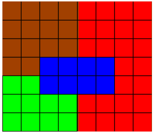
Dual graphs: properties

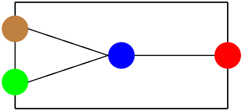
- If the vertices of G encode the regions then the vertice of $\overline{G}$ encode the intersection of borders (Nodes) and vice versa.
- Edges encode the borders (segments) of the partition.



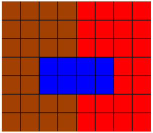


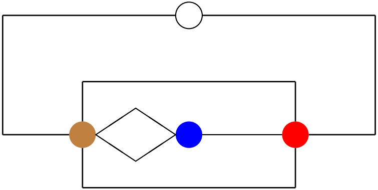
Characterising double edges



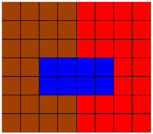


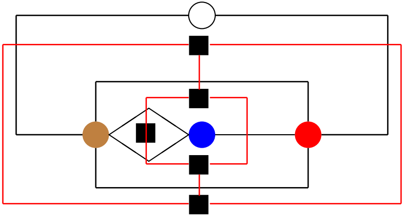
Characterising double edges





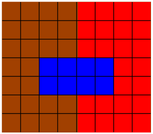
Characterising double edges

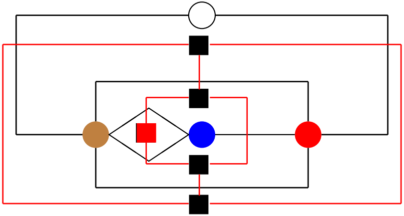




Characterising double edges

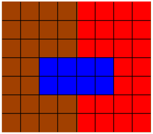
- A double edge is said to be redundant if it belongs to a dual vertex of degree 2.

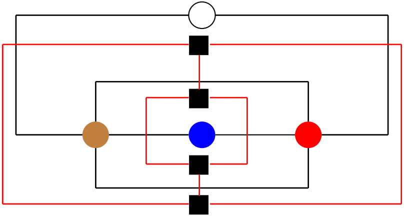




Characterising double edges

- A double edge is said to be redundant if it belongs to a dual vertex of degree 2.

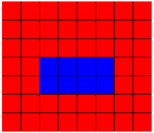


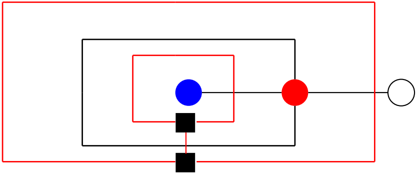


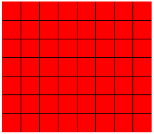
- We should thus remove all degree 2 vertices of $\overline{G}$.

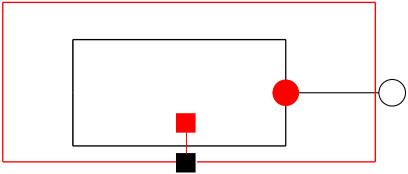
Processing of loops

- A loop is said to be redundant if it defines a dual vertex of degree 1.



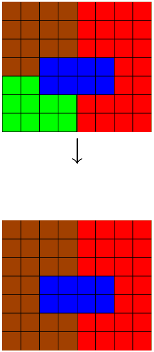


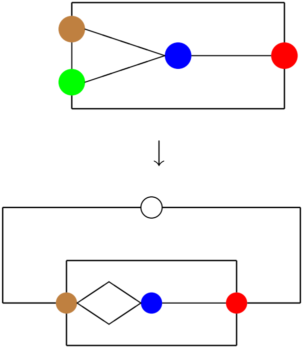




Merging two regions

- Contract in G one of the edge encoding the adjacency between both regions,
- Remove the corresponding edge in $\overline{G}$,

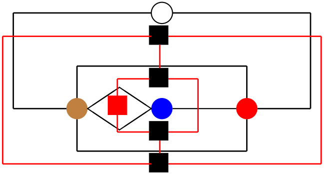




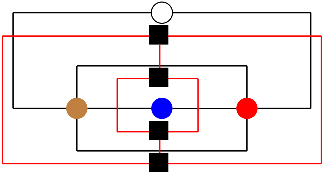
- Contract in $\overline{G}$ one of the two edges incident to vertice f such that $d(f) \leq 2$.

Merging two regions

- Contract in G one of the edge encoding the adjacency between both regions,
- Remove the corresponding edge in $\overline{G}$,
- Contract in $\overline{G}$ one of the two edges incident to vertice f such that $d(f) \leq 2$.



$\longrightarrow$



- Remove corresponding edges in G(loops, double edges).

Merging two regions

- Contract in G one of the edge encoding the adjacency between both regions,
- Remove the corresponding edge in $\overline{G}$,
- Contract in $\overline{G}$ one of the two edges incident to vertice f such that $d(f) \leq 2$.
- Remove corresponding edges in G(loops, double edges).

Merging two regions

	Dual Graphs	Simple Graphs (RAG)
Step 1	edge contraction	edge contraction
Step 2	removal of loops surrounding $f \in \overline{V}$ such that $d(f) = 1$	removal of all loops
Step 3	removal of double edges surrounding $f \in \overline{V}$ such that $d(f) = 2$	removal of all double edges

Conclusion

- ☹ Simple and dual graphs are essentially built using a bottom-up construction scheme.
- Compared to simple graphs, dual graphs allow to:
  - ☺ associate one edge to each connected boundary between 2 regions (segment),
  - ☺ characterise inside relationship.
- But:
  - ☹ Dual graphs do not fully use the properties of the plane embedding.
  - ☹ Does not allow to characterize locally inside relationships.

Conclusion

- ☹ Simple and dual graphs are essentially built using a bottom-up construction scheme.
- Compared to simple graphs, dual graphs allow to:
  - ☺ associate one edge to each connected boundary between 2 regions (segment),
  - ☺ characterise inside relationship.
- But:
  - ☹ Dual graphs do not fully use the properties of the plane embedding.
  - ☹ Does not allow to characterize locally inside relationships.

Conclusion

- ☹ Simple and dual graphs are essentially built using a bottom-up construction scheme.
- Compared to simple graphs, dual graphs allow to:
  - ☺ associate one edge to each connected boundary between 2 regions (segment),
  - ☺ characterise inside relationship.
- But:
  - ☹ Dual graphs do not fully use the properties of the plane embedding.
  - ☹ Does not allow to characterize locally inside relationships.

Conclusion

- What we may want:
  1. A model that may be built either bottom-up or top-down,
  2. which use a single data structure,
  3. which provide a local characterisation of inside relationships.
- Does it exists such a model ???
  - ☺ Yes.

Conclusion

- What we may want:
  1. A model that may be built either bottom-up or top-down,
  2. which use a single data structure,
  3. which provide a local characterisation of inside relationships.
- Does it exists such a model ???
  - ☺ Yes.

Conclusion

- What we may want:
  1. A model that may be built either bottom-up or top-down,
  2. which use a single data structure,
  3. which provide a local characterisation of inside relationships.
- Does it exists such a model ???
  - ☺ Yes.

Conclusion

- What we may want:
  1. A model that may be built either bottom-up or top-down,
  2. which use a single data structure,
  3. which provide a local characterisation of inside relationships.
- Does it exists such a model ???
  - ☺ Yes.

Conclusion

- What we may want:
  1. A model that may be built either bottom-up or top-down,
  2. which use a single data structure,
  3. which provide a local characterisation of inside relationships.
- Does it exists such a model ???
  - ☺ Yes.

Combinatorial maps

- Basic defs
  - Set D
  - Permutation : bijective application from D to D
    - Orbit of b in D according to π

$$< \pi > (b) = \{b, \pi(b), \pi^2(b), \dots, \pi^n(b)\}$$

      with $n \leq |D|$.
    - Cycles Decomposition: $\pi^*(b)$ restriction of π to $< \pi > (b)$ is a permutation from $< \pi > (b)$ to $< \pi > (b)$.

$$\pi = \pi^*(b_1) \dots, \pi^*(b_p)$$

Combinatorial maps

- Basic defs
  - Set D
  - Permutation : bijective application from D to D
    - Orbit of b in D according to π

$$< \pi > (b) = \{b, \pi(b), \pi^2(b), \ldots, \pi^n(b)\}$$

with $n \leq |D|$.

    - Cycles Decomposition: $\pi^*(b)$ restriction of π to $< \pi > (b)$ is a permutation from $< \pi > (b)$ to $< \pi > (b)$.

$$\pi = \pi^*(b_1) \ldots, \pi^*(b_p)$$

Combinatorial maps

- Basic defs
  - Set D
  - Permutation : bijective application from D to D
    - Orbit of b in D according to π

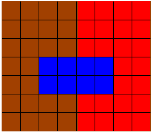
      $$< \pi > (b) = \{b, \pi(b), \pi^2(b), \dots, \pi^n(b)\}$$

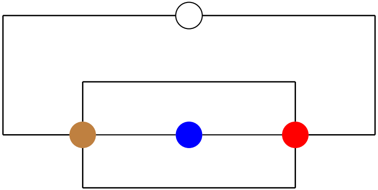
      with $n \leq |D|$.
    - Cycles Decomposition: $\pi^*(b)$ restriction of π to $< \pi > (b)$ is a permutation from $< \pi > (b)$ to $< \pi > (b)$.

      $$\pi = \pi^*(b_1) \dots, \pi^*(b_p)$$

Combinatorial maps: Edges

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

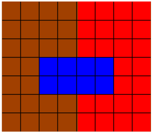


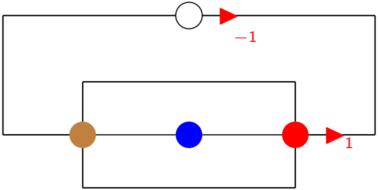


- Each edge is decomposed in two half edges called darts.
- Two darts of a same edge are connected by an involution α : $\alpha(1) = -1, \alpha(-1) = 1$

Combinatorial maps: Edges

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

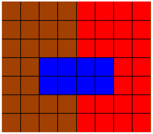


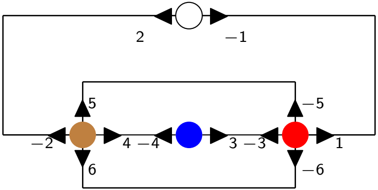


- Each edge is decomposed in two half edges called darts.
- Two darts of a same edge are connected by an involution α : $\alpha(1) = -1, \alpha(-1) = 1$

Combinatorial maps: Edges

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

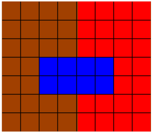


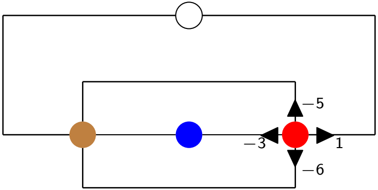


$$\mathcal{D} = \{-6, \dots, -1, 1, \dots, 6\}$$
$$\forall b \in \mathcal{D} \ \alpha(b) = -b$$

$$\alpha = (1, -1)(2, -2)(3, -3)(4, -4)(5, -5)(6, -6)$$

Vertices

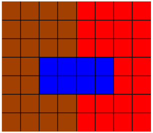


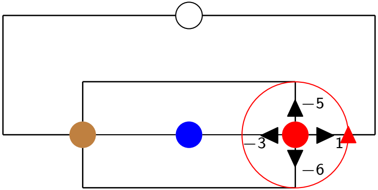


- Vertices are encoded by the cycles of σ.
- $\sigma^*(b)$ encode the sequence of darts encountered by turning with a positive orientation around the vertex containing b.

$$\sigma^*(1) = (1, -5, -3, -6)$$

Vertices

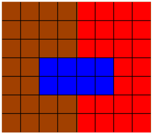


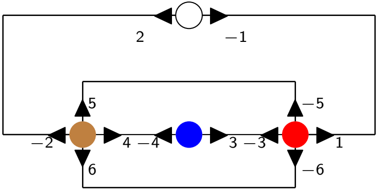


- Vertices are encoded by the cycles of σ.
- $\sigma^*(b)$ encode the sequence of darts encountered by turning with a positive orientation around the vertex containing b.

$$\sigma^*(1) = (1, -5, -3, -6)$$

Vertices



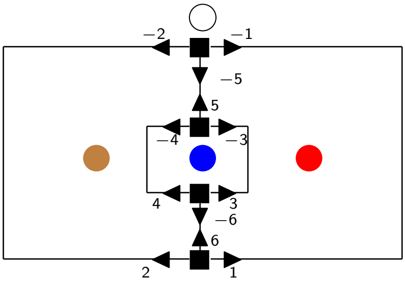


$$\sigma = (1, -5, -3, -6)(6, 4, 5, -2)(2, -1)(3, -4)$$

Dual combinatorial map

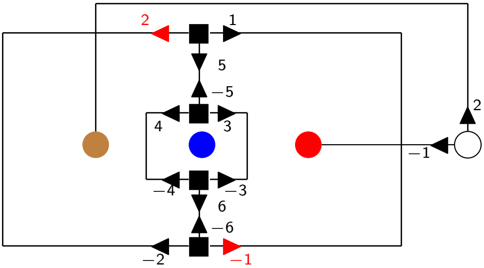
- Si $G = (\mathcal{D}, \sigma, \alpha)$ alors $\overline{G} = (\mathcal{D}, \varphi = \sigma \circ \alpha, \alpha)$.
- Les cycles de φ codent les faces de la carte duale (et donc la carte duale).

$$\varphi = (-2, -1, -5)(-4, 5, -3)(4, 3, -6)(2, 6, 1)$$



Infinite faces

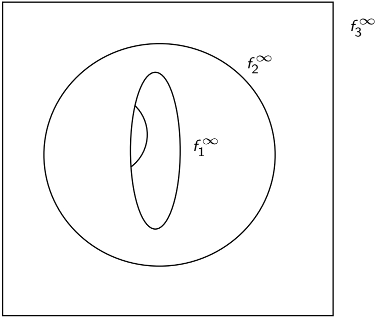
- If σ follows the positive orientation, all cycles of φ(faces) **but one** are traversed with the negative (clockwise) orientation.
- The cycle of φ traversed with a positive orientation is called the *Infinite face*. It encodes the complement of the connected component encoded by the combinatorial map. (Vertex ◯). The other faces are qualified of *finite* by reference to the domain they're surrounding.



$\varphi = (1, -6, -3, -5)(3, -4)(-2, 5, 4, 6)(-1, 2)$

Infinite faces

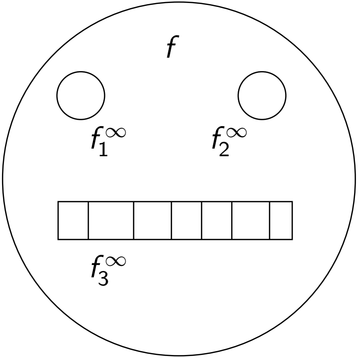
- We have one infinite face per connected component



- We must encode the inside relationships between these components.

An explicit encoding of inside relationships

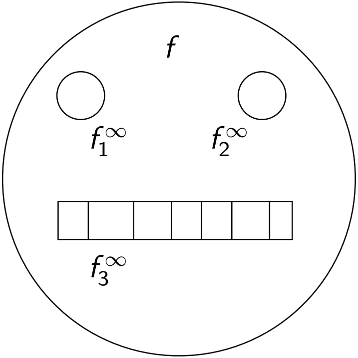
- **for any finite face** f **:**, $fille(f)$ infinite faces inside f
- **For any infinite face** f^∞ **:**$mere(f^\infty)$ finite face which contains it (limits its domain).



$$\begin{aligned} fille(f) &= \{f_1^\infty, f_2^\infty, f_3^\infty\} \\ mere(f_1^\infty) &= mere(f_2^\infty) \\ &= mere(f_3^\infty) \\ &= f \end{aligned}$$

An explicit encoding of inside relationships

- **for any finite face** f **:**, $fille(f)$ infinite faces inside f
- **For any infinite face** f^∞ **:**$mere(f^\infty)$ finite face which contains it (limits its domain).



$$
\begin{aligned}
fille(f) &= \{f_1^\infty, f_2^\infty, f_3^\infty\} \\
mere(f_1^\infty) &= mere(f_2^\infty) \\
&= mere(f_3^\infty) \\
&= f
\end{aligned}
$$

An explicit encoding of inside relationships

- **for any finite face** f **:**, $\textit{fille}(f)$ infinite faces inside f
- **For any infinite face** f^∞ **:**$\textit{mere}(f^\infty)$ finite face which contains it (limits its domain).
- ⚠ : The location of a newly Inserted connected component is not handled by the combinatorial map model ⇒ Requires geometrical information ⇒ Combination Combinatorial maps/geometrical models.

http://www.greyc.ensicaen.fr/ luc/ARTICLES/ecole_d_ete2.odp

Combinatorial maps: conclusion

- Implicit encoding of the dual
- Explicit encoding of the orientation
- Associated to inter-pixel boundaries, maps allow to:
  - Efficient updates of the partition encoding under split and merge operations,
  - Explicit encoding of inside relationships,
  - Efficient access to both geometrical and topological information

http://www.greyc.ensicaen.fr/∼luc/ARTICLES/ecole_d_ete2.odp

- May be extended to higher dimensions (3D,4D,. . . nD) at the price of a much higher memory cost.