



Graphs for image processing, analysis and pattern recognition

Florence Tupin

Overview

1. Definitions and representation models

2. Single graph methods

- ▶ Segmentation or labeling and graph-cuts
- ▶ Graphs for pattern recognition

3. Graph matching

- ▶ Graph or subgraph isomorphisms
- ▶ Error tolerant graph-matching
- ▶ Approximate algorithms (*inexact matching*)

Overview

1. Definitions and representation models

2. Single graph methods

- ▶ Segmentation or labeling and graph-cuts
- ▶ Graphs for pattern recognition

3. Graph matching

- ▶ Graph or subgraph isomorphisms
- ▶ Error tolerant graph-matching
- ▶ Approximate algorithms (*inexact matching*)

Why using graphs ?

- ▶ Interest: they give a compact, structured and complete representation, easy to handle
- ▶ Applications:
 - ▶ Image processing: segmentation, boundary detection
 - ▶ Pattern recognition: printed characters, objects (buildings 2D ou 3D, brain structures, ...), faces, ...
 - ▶ Image registration
 - ▶ Understanding of structured scenes
 - ▶ ...

Definitions

$$\text{Graph : } G = (X, E)$$

- ▶ X set of nodes ($|X|$ **order** of the graph)
- ▶ E set of edges ($|E|$ **size** of the graph)
- ▶ **complete** graph (size $\frac{n(n-1)}{2}$)
- ▶ **partial** graph $G = (X, E')$ with E' part of E
- ▶ **subgraph** $F = (Y, E')$, $Y \subseteq X$ et $E' \subseteq E$
- ▶ **degree** of a node x : $d(x) =$ number of edges
- ▶ **connected** graph: for each pair of nodes you find a path linking them
- ▶ **tree**: connected graph without cycle
- ▶ **clique**: complete subgraph
- ▶ **dual** graph (face \rightarrow node)
- ▶ **segment** graph (edge \rightarrow node)
- ▶ **hypergraph** (n-ary relations)
- ▶ **weighted** graphs: weights on the edges

Notations

Graph : $G = (X, E)$

- ▶ weight of an edge linking i et j : w_{ij}
- ▶ adjacency matrix W of size $|X| \times |X|$ defined by

$$W_{ij} = \begin{cases} w_{ij} & \text{if } e_{ij} \in E \\ 0 & \text{else} \end{cases}$$

for undirected edges W is symetric

- ▶ Laplacian matrix of an undirected graph

$$d_i = \sum_{e_{ij} \in E} w_{ij}$$

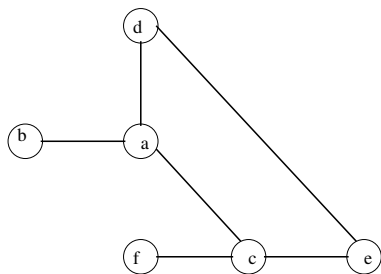
$$L_{ij} = \begin{cases} d_i & \text{if } i = j \\ -w_{ij} & \text{if } e_{ij} \in E \\ 0 & \text{else} \end{cases}$$

$$L = D - W$$

with $D_{ii} = d_i$ (D degree matrix)

Representation

Adjacency matrix, adjacency lists



	a	b	c	d	e	f
a	0	1	1	1	0	0
b	1	0	0	0	0	0
c	1	0	0	0	1	1
d	1	0	0	0	1	0
e	0	0	1	1	0	0
f	0	0	1	0	0	0

Representation

Adjacency matrix, adjacency lists

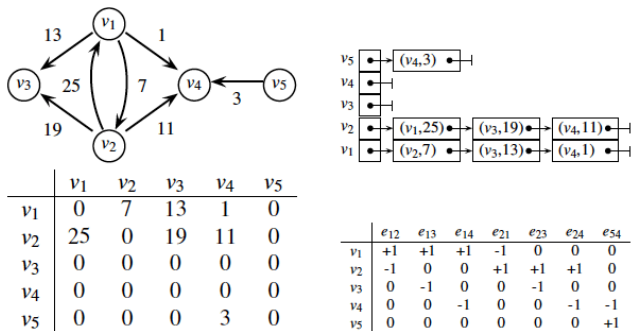


FIGURE 1.4

From top-left to bottom-right: a weighted directed graph, its adjacency list, its adjacency matrix, and its (transposed) incidence matrix representations.

(figure from “Image processing and analysis with graphs”, Lezoray - Grady)

Which graphs for images ?

Examples of graphs

- ▶ **Attributed graph** : $G = (X, E, \mu, \nu)$
 - ▶ $\mu : X \rightarrow L_X$ nodes interpreter ($L_X =$ attributes of nodes)
 - ▶ $\nu : E \rightarrow L_E$ edges interpreter ($L_E =$ attributes of edges)

Exemples :

- ▶ graph of pixels
- ▶ region adjacency graph (RAG)
- ▶ Voronogions / Delaunay triangulation
- ▶ graph of primitives with complex relationships
- ▶ **Random graph** : edges and nodes = random variables
- ▶ **Fuzzy graph** : $G = (X, E = X \times X, \mu_f, \nu_f)$
 - ▶ $\mu_f : X \rightarrow [0, 1]$
 - ▶ $\nu_f : E \rightarrow [0, 1]$
 - ▶ avec $\forall (u, v) \in X \times X \quad \nu_f(u, v) \leq \mu_f(u)\mu_f(v)$ or $\nu_f(u, v) \leq \min[\mu_f(u)\mu_f(v)]$

Examples of image graphs

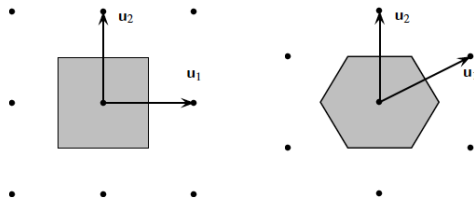


FIGURE 1.11

The rectangular (left) and hexagonal (right) lattices and their associated Voronoi cells.

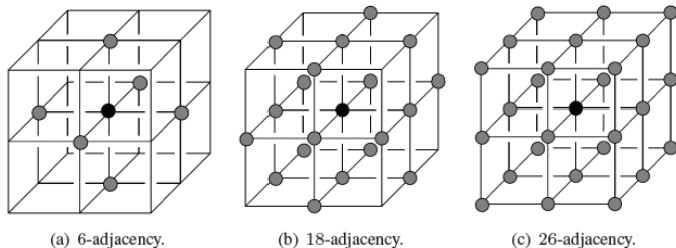
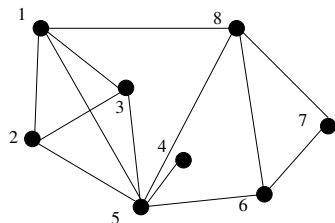
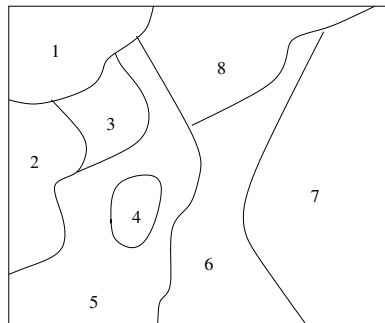


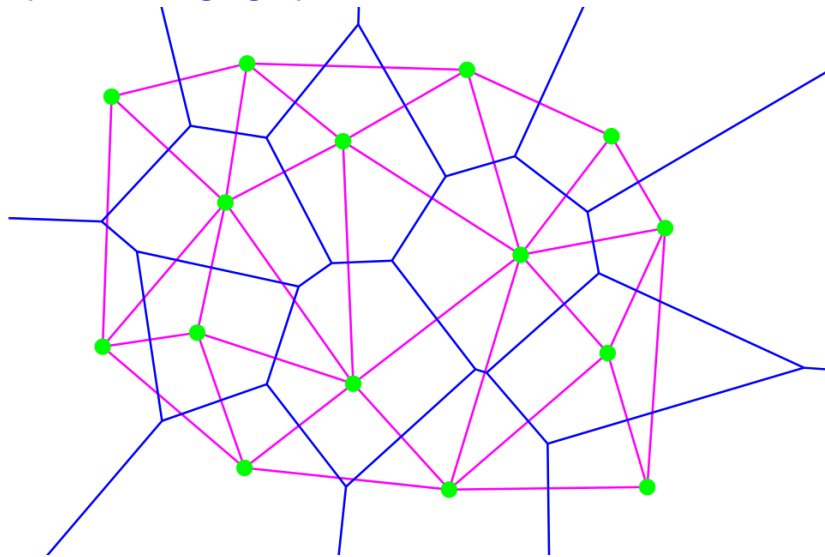
FIGURE 1.12

Examples of image graphs



RAG (Region Adjacency Graph)

Examples of image graphs



Voronoi diagram (in blue) and Delaunay triangulation (pink)

Examples of image graphs

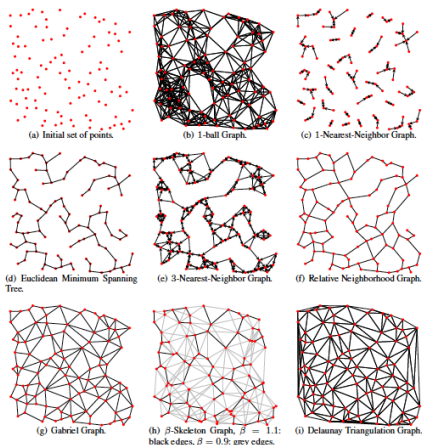


FIGURE 1.14

Examples of proximity graphs from a set of 100 points in \mathbb{Z}^2 .

(figure from “Image processing and analysis with graphs”, Lezoray
- Grady)

Examples of graphs

- ▶ **Graph of fuzzy attributes** : attributed graph with fuzzy value for each attribute
- ▶ **Hierarchical graph** :
multi-level graph and and bi-partite graph between 2 levels
(multi-level approaches, object grouping, ...)

Exemples :

- ▶ quadtrees, octrees
- ▶ hierarchical representation of the brain
- ▶ **Graph for reasoning**
decision tree, matching graph

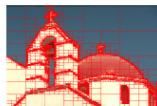
Graph example

0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

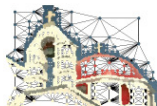
(a)



(b)



(c)



(d)



(e)



(f)

FIGURE 1.13

(a) An image with the quadtree tessellation, (b) the associated partition tree, (c) a real image with the quadtree tessellation, (d) the region adjacency graph associated to the quadtree partition, (e) and (f) two different irregular tessellations of an image using image-dependent superpixel segmentation methods: Watershed [23] and SLIC superpixels [24].

(figure from “Image processing and analysis with graphs”, Lezoray
- Grady)

Graph examples

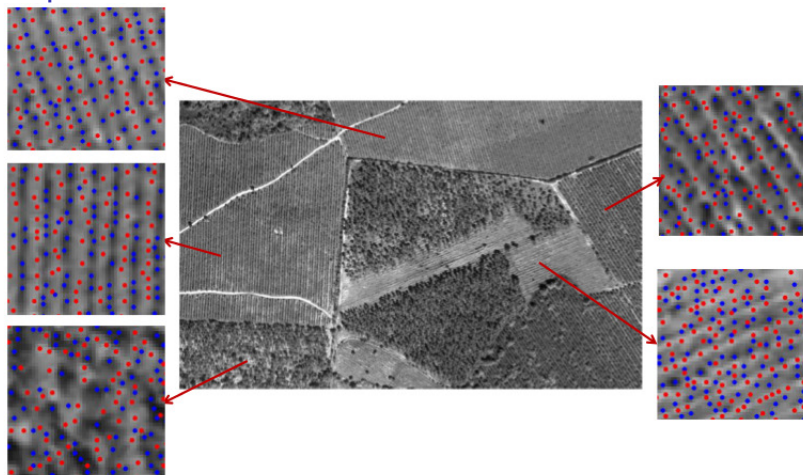
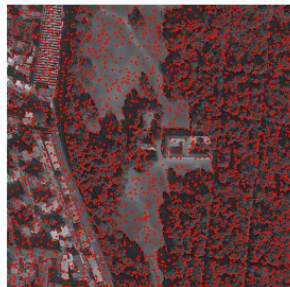


Figure 2 – Représentation de variété des points clés de $\mathcal{S}_\omega^{\max}(I)$ (en rouge) et $\mathcal{S}_\omega^{\min}(I)$ (en bleu) image Pléiades ayant des textures locales différentes.

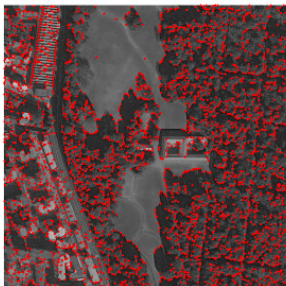
(figure from M.T. Pham PhD)



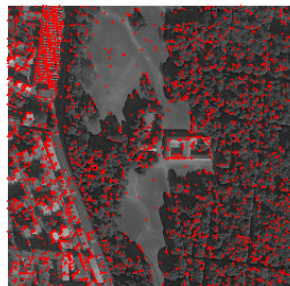
(a) Image initiale 512×512



(b) Extrema locaux



(c) Détecteur de Harris



(d) Détecteur SIFT

Graph examples

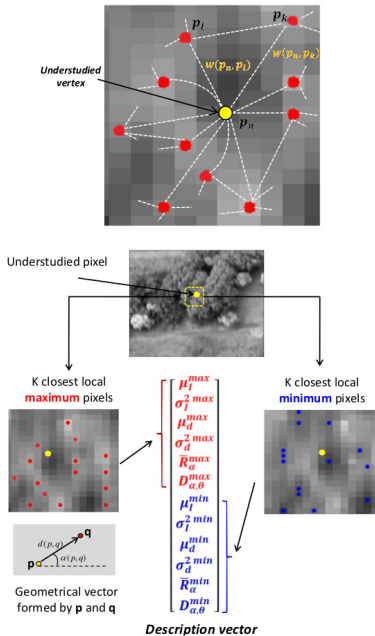
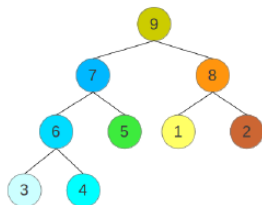
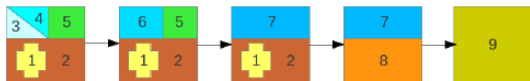


Figure 5 – Vecteur de description proposé pour l'analyse ponctuelle de la texture.

(figure from M.T. Pham PhD)

Graph examples - BPT Binary Partition Tree



Which algorithms from graph theory ?

Some classical algorithms

Search of the minimum spanning tree

- ▶ Kruskal algorithm $O(n^2 + m \log_2(m))$
- ▶ Prim algorithm $O(n^2)$

Shortest path problems

- ▶ positive weights: Dijkstra algorithm $O(n^2)$
- ▶ arbitrary weights but without cycle: Bellman algorithm $O(n^2)$

Max flow and Min cut

- ▶ $G = (X, E)$
- ▶ partitioning in two sets A et B ($A \cup B = X$, $A \cap B = \emptyset$)
- ▶ $cut(A, B) = \sum_{x \in A, y \in B} w(x, y)$
- ▶ Ford and Fulkerson algorithm

Search of maximal clique in a graph

- ▶ decision tree
- ▶ cut of already explored branches

Overview

1. Definitions and representation models

2. Single graph methods

- ▶ Segmentation or labeling and graph-cuts
- ▶ Graphs for pattern recognition

3. Graph matching

- ▶ Graph or subgraph isomorphisms
- ▶ Error tolerant graph-matching
- ▶ Approximate algorithms (*inexact matching*)

Segmentation by minimum spanning tree

How can we segment this image using a minimum spanning tree ?



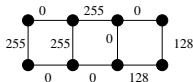
Segmentation by minimum spanning tree

Constantinides (1986)

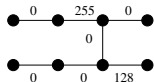
- ▶ graph of pixels weighted by the gray levels (or colors) (weights = distances)
- ▶ search of the minimum spanning tree
- ▶ spanning tree \Rightarrow partitioning by suppressing the most costly edges



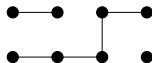
image



graphe des pixels attribué



arbre couvrant de poids minimal



suppression des arêtes
les plus coûteuses

Computation of the minimum spanning tree

Kruskal algorithm

- ▶ Starting from a partial graph without any edge, iterate $(n - 1)$ times : choose the edge of minimum weight creating no cycle in the graph with the previously chosen edges
- ▶ In practice:
 1. sorting of edges by increasing weights
 2. while the number of edges is less than $(n - 1)$ do:
 - ▶ select the first edge not already examined
 - ▶ if cycle, reject
 - ▶ else, add the edge in the graph
- ▶ Complexity: $O(n^2 + m \log_2(m))$

Prim algorithm

- ▶ Extension from near to near of the current tree
- ▶ Complexity: $O(n^2)$

Constantinides (1986)



a



b



c



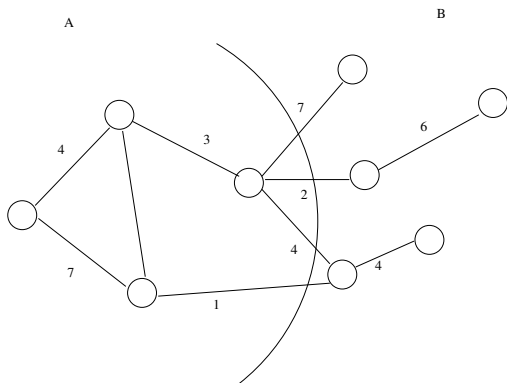
new boundary

d

Segmentation by graph-cut

Graph-cut definition:

- ▶ graph $G = (X, E)$
- ▶ partitioning in 2 parts A et B ($A \cup B = X, A \cap B = \emptyset$)
- ▶ $cut(A, B) = \sum_{i \in A, j \in B} w_{ij}$



Segmentation by graph clustering

Clustering : partitioning of the graph in groups of nodes based on their similarities [Each cluster : a closely connected component]

The clustering corresponds to:

- ▶ edges between different groups have low weights (weak similarities)
- ▶ edges inside a group have high weights (high similarities)

Possible cost functions for the cut:

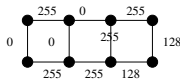
- ▶ minimum cut $Cut(A_1, \dots, A_k) = \sum_{i=1}^{i=k} Cut(A_i, \bar{A}_i)$
- ▶ minimum cut normalized by the size of each part (RatioCut)
 $RatioCut(A_1, \dots, A_k) = \sum_{i=1}^{i=k} \frac{1}{|A_i|} Cut(A_i, \bar{A}_i)$
($|A_i|$ number of vertices in A_i)
- ▶ minimum cut normalized by the connectivity of each part (NCut)
 $NCut(A_1, \dots, A_k) = \sum_{i=1}^{i=k} \frac{1}{vol(A_i)} Cut(A_i, \bar{A}_i)$
($vol(A_i) = \sum_{k \in A_i} d_k$ sum of the weight of all edges of vertices in A_i)

Toy example

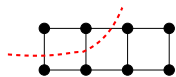
Wu and Leavy (93): search for the MinCut



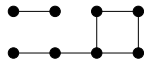
image



graphe des pixels attribué

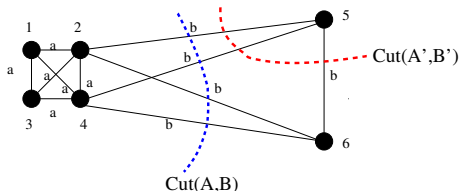


coupe de capacité minimale



partition

Influence of the number of edges: $Cut(A, B) = 4b$,
 $Cut(A', B') = 3b$



⇒ normalized cut (NCut)

Normalized cut

- ▶ Principle: graph clustering
- ▶ + suppression of the influence of the number of edges:
normalized cut

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, X)} + \frac{cut(A, B)}{assoc(B, X)}$$

$$assoc(A, X) = \sum_{a \in A, x \in X} w(a, x) = vol(A)$$

- ▶ Measuring the connectivity of a cluster:

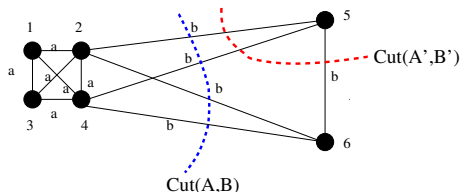
$$Nassoc(A, B) = \frac{assoc(A, A)}{assoc(A, X)} + \frac{assoc(B, B)}{assoc(B, X)}$$

$$Ncut(A, B) = 2 - Nassoc(A, B)$$

minimizing the cut \Leftrightarrow maximizing group connectivity

Toy example

Influence of the number of edges: $Cut(A, B) = 4b$,
 $Cut(A', B') = 3b$



\Rightarrow normalized cut (NCut)

$vol(A) =$

$vol(B) =$

$NCut(A, B) =$

$vol(A') =$

$vol(B') =$

$NCut(A', B') =$

Graph theory and cuts

MinCut by combinatorial optimization

- ▶ Stoer-Wagner algorithm
- ▶ Principle: iterative reducing of the graph by fusion of the nodes linked by the maximal weights

Min K-cut by combinatorial optimization

- ▶ Partitioning the (un-oriented graph) graph in many components
- ▶ Gomory-Hu algorithm

minCut in oriented graph by combinatorial optimization

- ▶ Ford-Fulkerson algorithm (oriented graph with two terminal nodes (sink / tank))
- ▶ Principle: MaxFlow search (MinCut equivalence) by search for an augmenting chain to increase the flow

Graph theory and cuts

Laplacian matrices

$D = \text{diag}(d_i)$ with $d_i = \sum_j w_{ij}$

$W = (w_{ij})$

- ▶ Graph Laplacian matrix

$$L = D - W$$

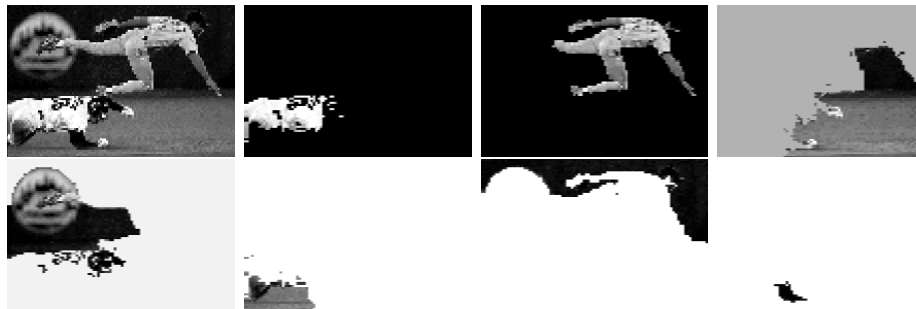
- ▶ Normalized graph Laplacian matrix

$$L_n = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

Spectral clustering algorithms and cuts

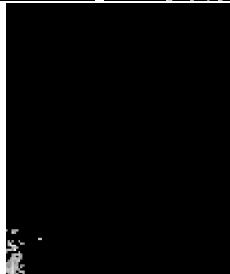
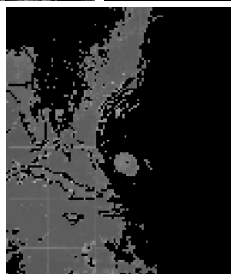
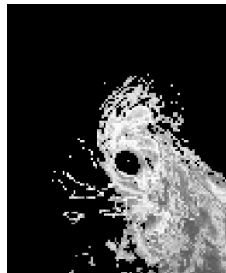
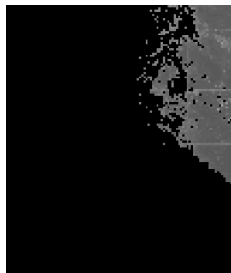
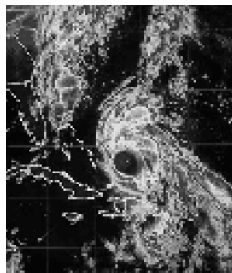
- ▶ Computation of the eigen-values and eigen-vectors of some matrix (L , L_n , or generalized eigen problems $Lu = \lambda Du$)
- ▶ selection of the k smallest eigen-values and associated k eigen-vectors u_k
- ▶ $U = (u_1, \dots, u_k) \in R^{n \times k}$
- ▶ let $y_i \in R^k$ be the i th row of U ($i = 1, \dots, n$)
- ▶ cluster the points $(y_i)_{1 \leq i \leq n}$ with the k -means algorithm into clusters C_1, \dots, C_k
- ▶ clusters A_1, \dots, A_k with $A_i = \{j | y_j \in C_i\}$

Examples (univ. Berkeley)



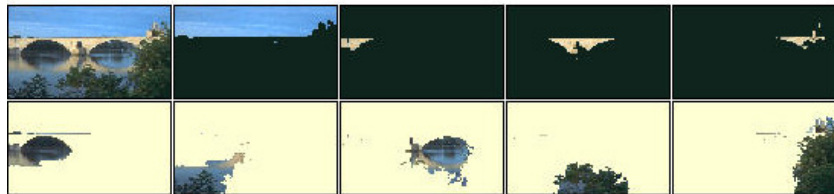
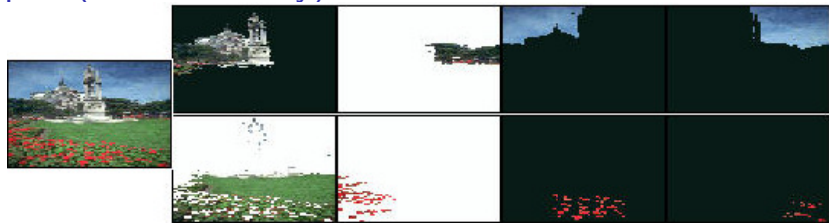
<http://www.cs.berkeley.edu/projects/vision/Grouping/>

Examples (univ. Berkeley)



<http://www.cs.berkeley.edu/projects/vision/Grouping/>

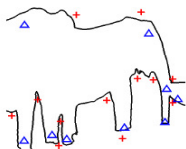
Examples (univ. Berkeley)



Examples (univ. Alberta) with linear constraints



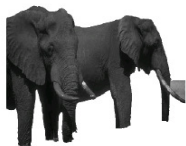
(a)



(b)



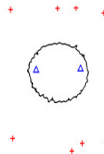
(c)



(d)



(a)



(b)



(c)

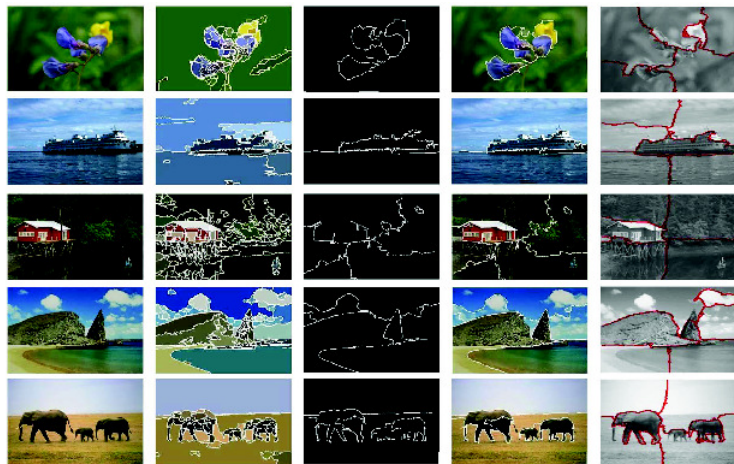


(d)



(e)

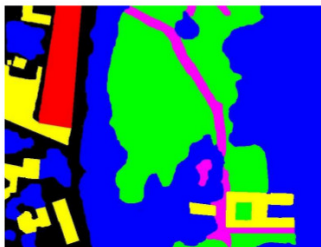
Examples (Mean Shift et Normalized Cut)



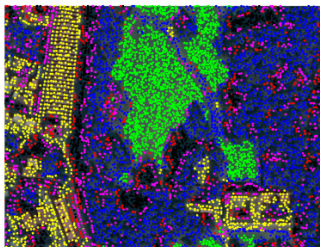
Examples (texture classification with point-wise graph)



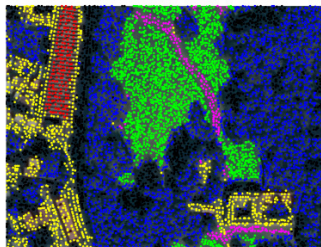
(a) Image originale



(b) Vérité terrain



(j) Vecteur de signature



(k) Classification spectrale des vecteurs de signature

Overview

1. Definitions and representation models

2. Single graph methods

- ▶ Segmentation or labeling and graph-cuts
- ▶ Graphs for pattern recognition

3. Graph matching

- ▶ Graph or subgraph isomorphisms
- ▶ Error tolerant graph-matching
- ▶ Approximate algorithms (*inexact matching*)

Full scene labeling (scene parsing)



Figure from Farabet et al., PAMI
Tenenbaum and Barrow (1977)

- ▶ Segmentation in regions
- ▶ Building of the Region Adjacency Graph
- ▶ Labeling using a set of rules (expert system) :
 1. on objects (size, color, texture,...)
 2. on contextual relationships between objects (above, inside, near ...)

Generalization with fuzzy attributed graphs

Markovian labeling (random graphs)

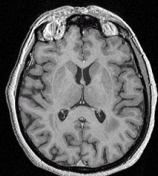
$$E(l) = \sum_i \Phi(d_i, l_i) + \beta \sum_{ij} \Psi(l_i, l_j)$$

- ▶ Low-level applications:
 - ▶ pixel graphs
 - ▶ segmentation, classification, restoration
- ▶ High-level applications:
 - ▶ graph of super-pixels (SLIC, watershed, ...)
 - ▶ graph of primitives (edges, key-points, lines,...)
- ▶ CRF (Conditional Random Field) / MRF (Markov Random Field):
 - ▶ MRF: Ψ does not depend on d ("pure" prior)
 - ▶ CRF: Ψ depends on d (usually based on image gradient values)

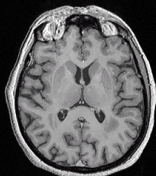
⇒ pattern recognition, full scene labeling

Example on a 3D RAG (T. Gud)

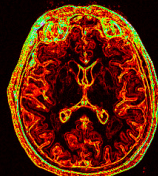
*nuclei
segmentation*



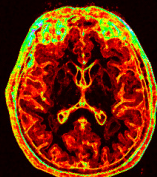
a data slice



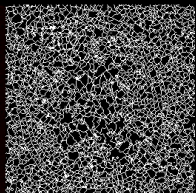
3D anisotropic diffusion



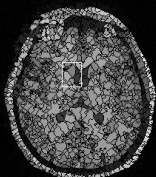
3D anisotropic gradient



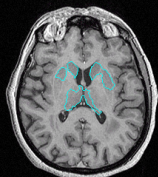
3D morphological closing



3D watershed

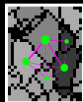


3D over-segmentation



result of graph labeling

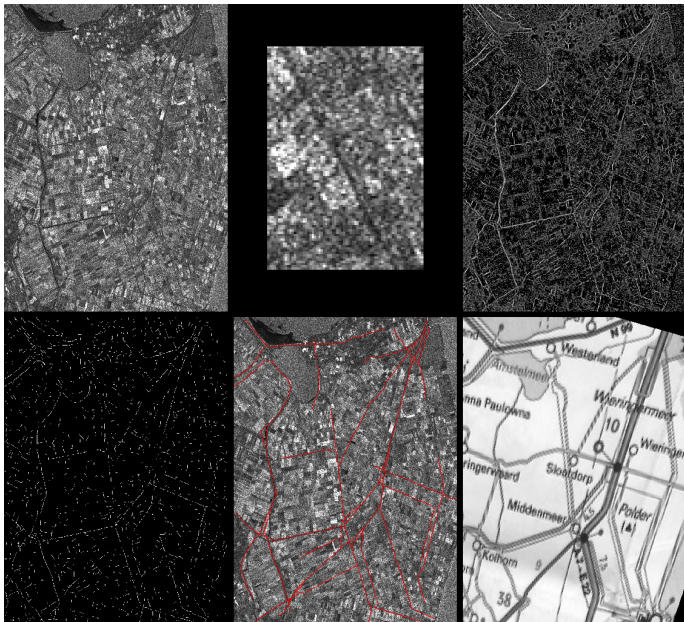
Markovian relaxation



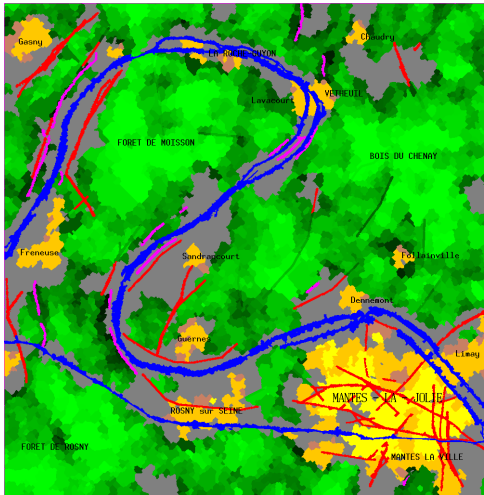
$$p(y|x) = \exp\left[-\sum_s \frac{\text{vol}_s (y_s - \mu_s)^2}{2\sigma_s^2}\right]$$

$$p(x) = \frac{1}{Z} \exp\left[-\sum_c \sum_{s \in \mathcal{N}(c)} \text{surf}_c P[x_s, x_n]\right]$$

Example on a line graph



Example on a region adjacency graph



MRF and graph-cut optimization

Binary labeling (Greig et al. 89) :

$$\mathcal{E}(l) = \sum_i \Phi(d_i | l_i) + \sum_{(i,j)} \beta(l_i - l_j)^2$$

- ▶ efficient way of finding the global minimum by min-cut search

MRF and graph-cut optimization

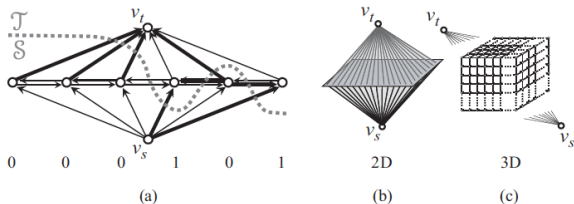


FIGURE 2.5

(a) The graph for binary MRF minimization. The edges in the cut are depicted as thick arrows. Each node other than v_s and v_t corresponds to a site. If a cut $(\mathcal{S}, \mathcal{T})$ places a node in \mathcal{S} , the corresponding site is labeled 0; if it is in \mathcal{T} , the site is labeled 1. The 0's and 1's at the bottom indicate the label each site is assigned. Here, the sites are arranged in 1D; but according to the neighborhood structure this can be any dimension as shown in (b) and (c).

(figure from “Image processing and analysis with graphs”, Lezoray - Grady)

MRF/CRF and graph-cut optimization

Multi-level labeling (Boykov, Veksler) :

⇒ generalization of the binary labeling

Definition of two space moves (to go back to the binary labeling)

- ▶ α -expansion : source S and sink P correspond to label α and the current label $\bar{\alpha}$ (Ψ should be a metric)
- ▶ $\alpha - \beta$ swap: source S for α and sink P for β (Ψ should be a semi-metric)

Optimization by iterative mincut search:

- ▶ graph: nodes for super-pixels
- ▶ weights: depending on the current labeling
- ▶ good trade off time / efficiency compared to simulated annealing or ICM

But for multi-labeling no guarantee on optimality of the solution

Interactive segmentation: “hard” constraints

Principle Background and object manually defined

⇒ finding of a binary labeling minimizing an energy including “hard” constraints

Method Mincut search and edges with high weights (should not be cut)

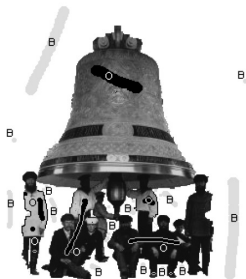
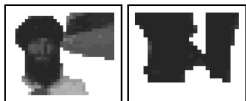
Advantages

- ▶ easy introduction of “hard” constraints
- ▶ the manually defined areas permit to do a fast learning
- ▶ iterative algorithm

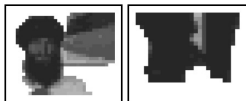
Illustrations



(a) Original B&W photo



(b) Segmentation results



Interactive methods with mincut

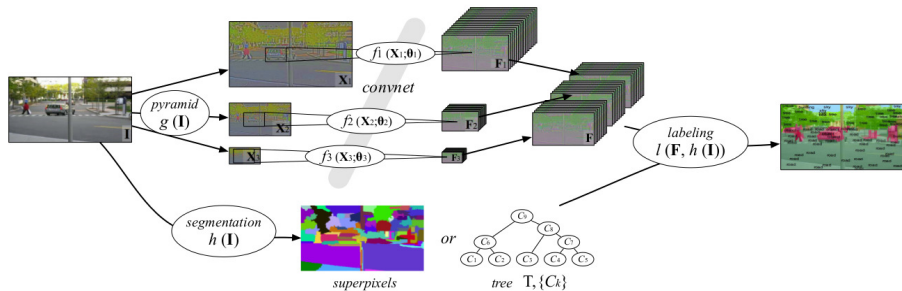
Grab-cut

- ▶ take into account color
- ▶ two labels (background and object but with a Gaussian Mixture Model)
- ▶ CRF (conditional random field): regularization term weighted by the image gradient
- ▶ iterative semi-supervised learning of the GMM parameters (after manual initialization and after each cut)

Illustrations -GrabCut-



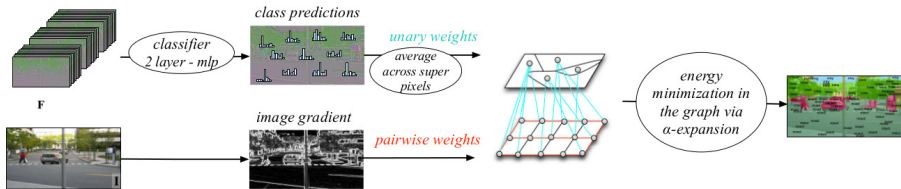
Deep learning and graph labeling for full scene labeling



Deep learning and graph labeling for full scene labeling

$$\Phi(d_i, l_i) = \exp(-\alpha d_{i,a}) 1(l_i \neq a)$$

$$\Psi(l_i, l_j) = \exp(-\beta \|\nabla I\|_i) 1(l_i \neq l_j)$$



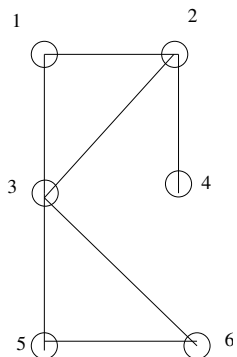
Pattern recognition

- ▶ Object: defined by a set of primitives (nodes of the graph)
- ▶ Binary relationship of compatibility between nodes (edges of the graph)
- ▶ Clique: sub-set of primitives all compatible between each other = possible object configuration
- ▶ recognition by maximal clique detection

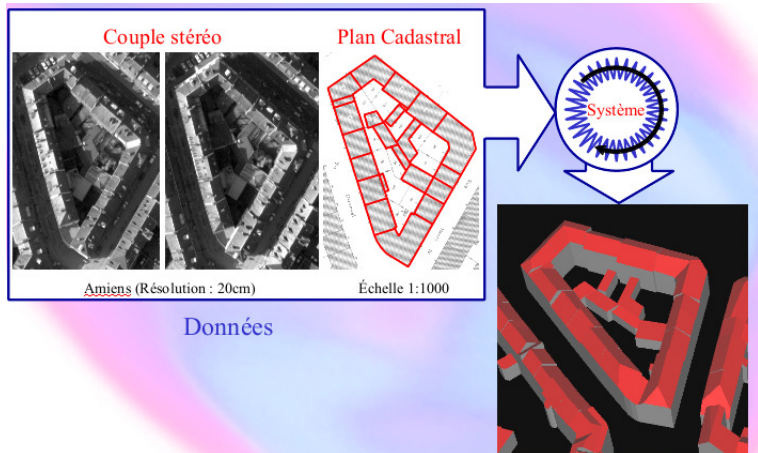
Search of maximal cliques :

- ▶ NP-hard problem
- ▶ Building of a decision tree: a node of the tree = 1 clique of the graph
- ▶ pruning of the tree to suppress already found cliques
- ▶ Theorem: let S be a node of the search tree T , and let x be the first unexplored child of S to be explored. If all the sub-trees of $S \cup \{x\}$ have been generated, only the sons S not adjacent to x have to be explored.

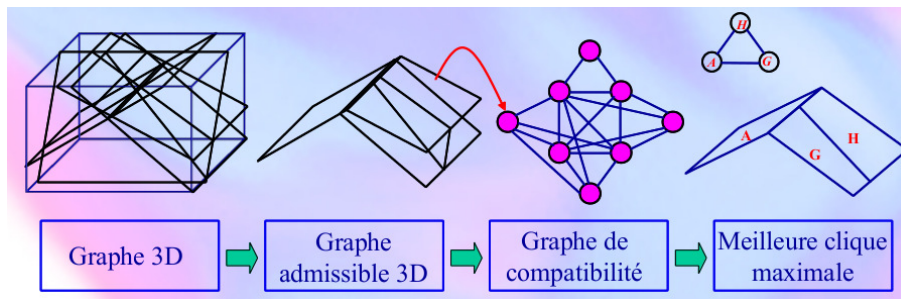
Example: maximal clique search



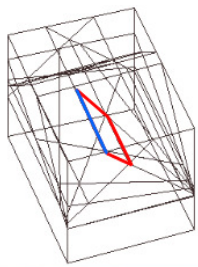
Example: buiding reconstruction by the maximal clique search (IGN)



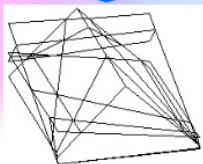
Example: building reconstruction by the maximal clique search (IGN)



Example: building reconstruction by the maximal clique search (IGN)



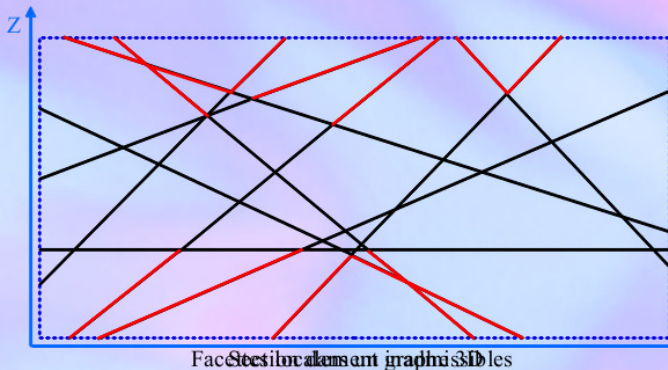
Graphe 3D



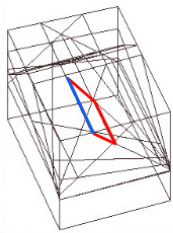
Graphe Admissible 3D

Algorithme :

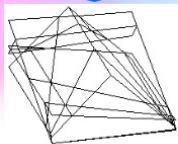
Éliminer récursivement toute facette localement inadmissible



Example: building reconstruction by the maximal clique search (IGN)

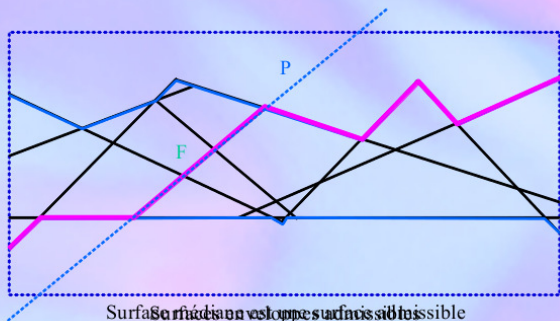


Graphe 3D

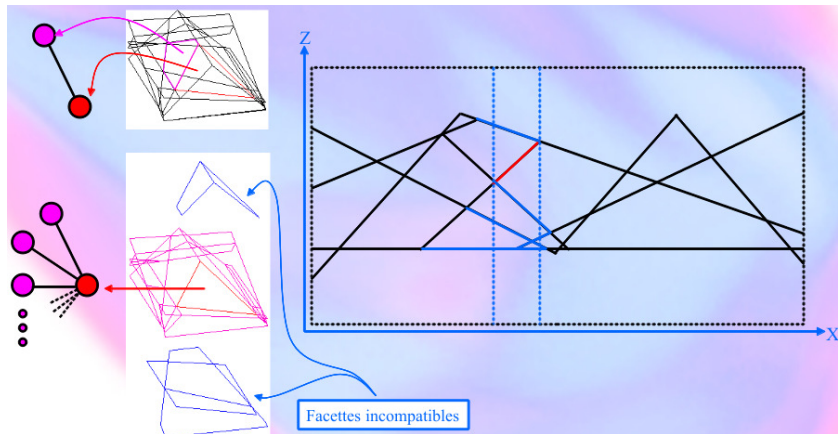


Algorithme :

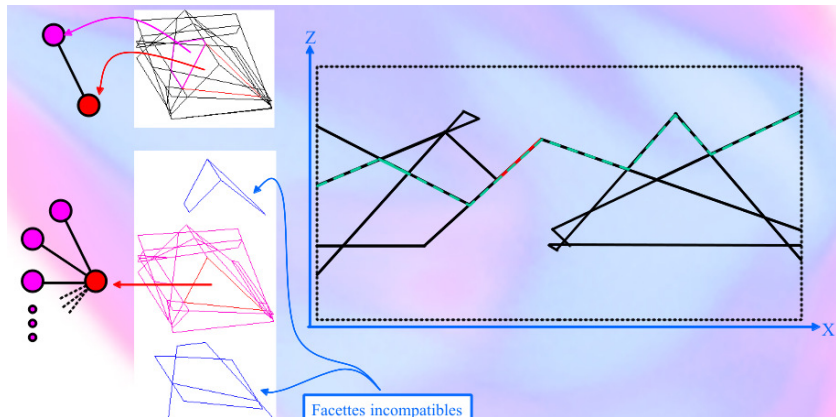
Éliminer récursivement toute facette localement inadmissible



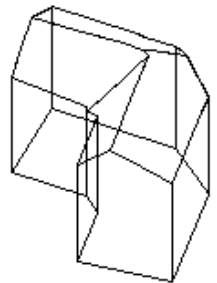
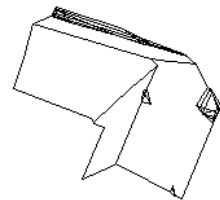
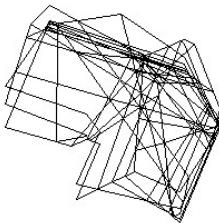
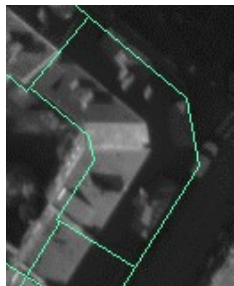
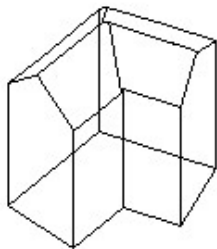
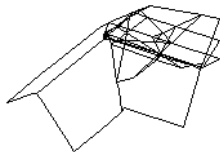
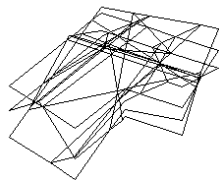
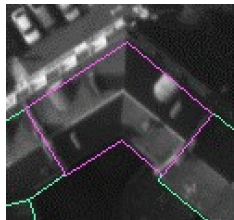
Example: building reconstruction by the maximal clique search (IGN)



Example: building reconstruction by the maximal clique search (IGN)



Example: buiding reconstruction by the maximal clique search (IGN)



Overview

1. Definitions and representation models

2. Single graph methods

- ▶ Segmentation or labeling and graph-cuts
- ▶ Graphs for pattern recognition

3. Graph matching

- ▶ Graph or subgraph isomorphisms
- ▶ Error tolerant graph-matching
- ▶ Approximate algorithms (*inexact matching*)

Graph matching

Correspondance problem:

- ▶ Graph(s) of the model (atlas, map, model of object)
- ▶ Graph built from the data
- ▶ Graph matching:

$$G = (X, E, \mu, \nu) \rightarrow? G' = (X', E', \mu', \nu')$$

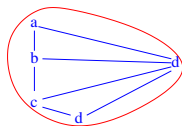
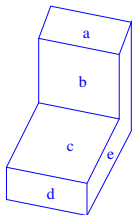
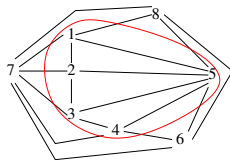
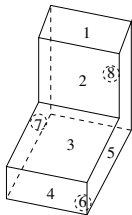
Graph isomorphism: bijective function $f : X \rightarrow X'$

- ▶ $\mu(x) = \mu'(f(x))$
- ▶ $\forall e = (x_1, x_2), \exists e' = (f(x_1), f(x_2)) / \nu(e) = \nu'(e')$ and conversely

Too strict \Rightarrow **isomorphisms of sub-graphs**

Sub-graph isomorphisms

- There exists a sub-graph S' of G' such that f is an isomorphism from G to S'

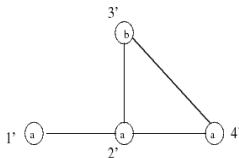
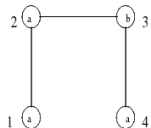


- There exists a sub-graph S of G and a sub-graph S' of G' such that f is an isomorphism from S to S'

Graph isomorphisms: searching the maximal clique

search of the **maximal clique of the association graph**

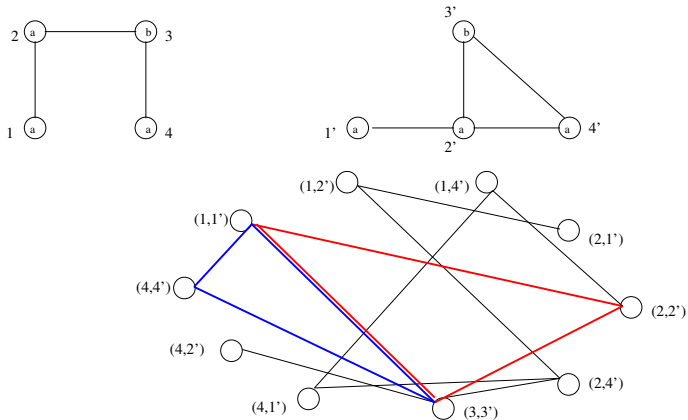
- ▶ principle: building of the association graph
- ▶ maximal clique: sub-graph isomorphism



Graph isomorphisms: searching the maximal clique

search of the **maximal clique of the association graph**

- ▶ principle: building of the association graph
- ▶ maximal clique: sub-graph isomorphism



Sub-graph isomorphism: Ullman algorithm

- ▶ Principle : extension of the association set (v_i, w_{x_i}) until the G graph has been fully explored. In case of failure, go back in the association graph (“backtrack”).
Acceleration: “forward checking” before adding an association.
- ▶ Algorithm:
 - ▶ matrix of node associations
 - ▶ matrix of future possible associations for a given set of associations matrice
 - ▶ list of updated associations by “Backtrack” et “ForwardChecking”
- ▶ Complexity : worst case $O(m^n n^2)$ (n ordre de X , m de X' , $n < m$)

Overview

1. Definitions and representation models

2. Single graph methods

- ▶ Segmentation or labeling and graph-cuts
- ▶ Graphs for pattern recognition

3. Graph matching

- ▶ Graph or subgraph isomorphisms
- ▶ Error tolerant graph-matching
- ▶ Approximate algorithms (*inexact matching*)

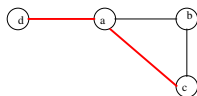
Error tolerant graph-matching

- ▶ Real world: noisy graphs, incomplete graphs, distortions
- ▶ Distance between graphs (editing, cost function,...)
- ▶ Sub-graph isomorphism with error tolerance: search of the sub-graph G' with the minimum distance to G
- ▶ Optimal algorithms: A^*
- ▶ Approximate matching: genetic algorithms, simulated annealing, neural networks, probabilistic relaxation,...
 - ▶ iterative minimization of an objective function
 - ▶ better adapted for big graphs
 - ▶ problem of convergence and local minima

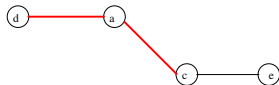
Decomposition in common sub-graphs

Messmer, Bunke

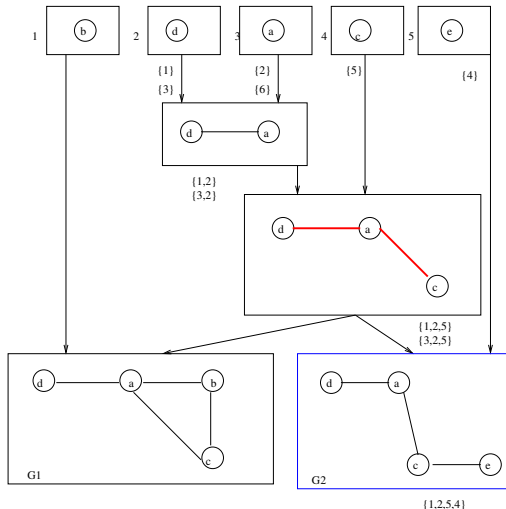
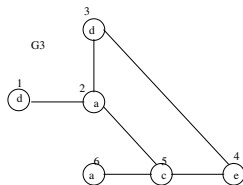
G1



G2

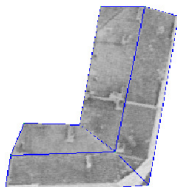
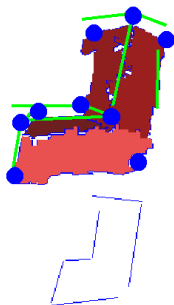


G3



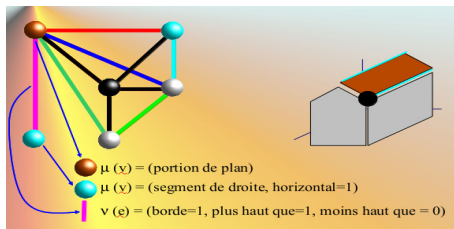
Example

3D reconstruction by graph matching between a graph (data) and a library of model graphs (IGN)



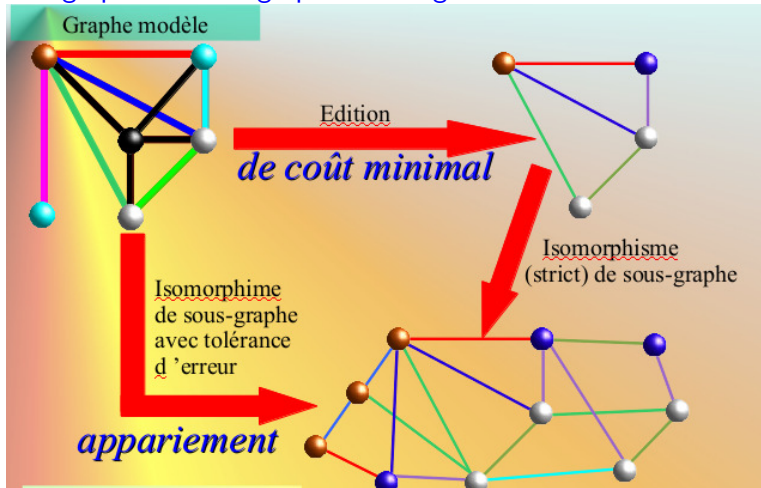
Example - building reconstruction

Model graph



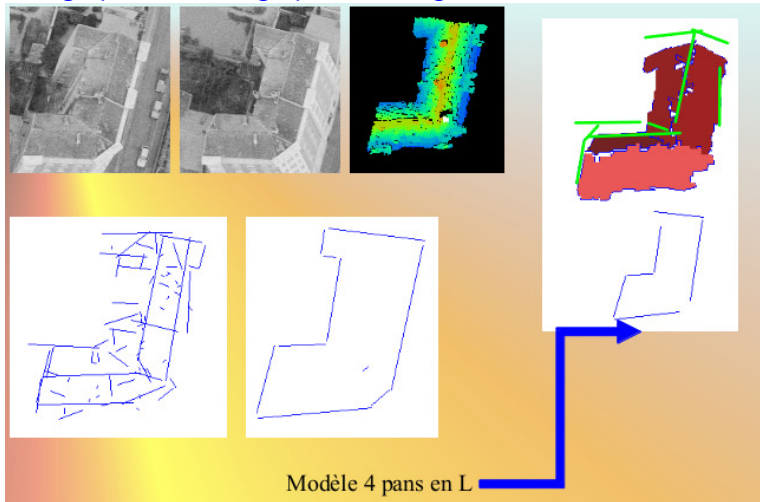
Example - building reconstruction

Model graph and data graph matching



Example - building reconstruction

Model graph and data graph matching



Overview

1. Definitions and representation models

2. Single graph methods

- ▶ Segmentation or labeling and graph-cuts
- ▶ Graphs for pattern recognition

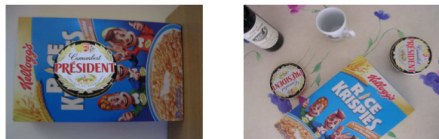
3. Graph matching

- ▶ Graph or subgraph isomorphisms
- ▶ Error tolerant graph-matching
- ▶ Approximate algorithms (*inexact matching*)

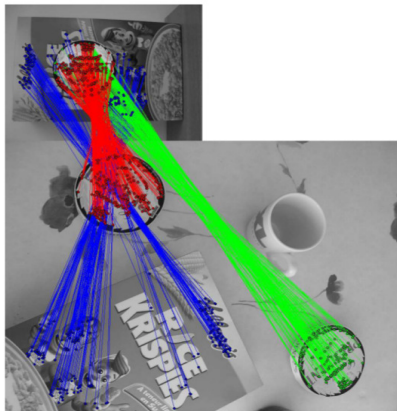
Matching with geometric transformation

- ▶ Graph = representation of the spatial information
- ▶ Matching = computation of the geometric transformation
 - ▶ polynomial deformation
 - ▶ elastic transformation (morphing)
- ▶ Matching approaches :
 - ▶ translation: maximum of correlation
 - ▶ Hough transform (in the parameter space)
 - ▶ RANSAC method: select randomly a set of matching points, compute the transformation, compute the score (depends on the number of matched pairs for the transformation)
 - ▶ AC-RANSAC: RANSAC + a contrario framework reducing the number of parameters (NFA to be set)

Example - MAC-RANSAC (PhD Julien Rabin)



(a) Paire d'images analysée.



(b) Reconnaissance de chacun des objets superposés.

Example - MAC-RANSAC (PhD Julien Rabin)



(a) Paire d'images utilisée



Inexact matching

Optimization of a cost function

- ▶ Dissimilarity cost between nodes

$$c_N(a_D, a_M) = \sum \alpha_i d(a_i^N(a_D), a_i^N(a_M)) \quad \sum \alpha_i = 1$$

- ▶ Dissimilarity cost between edges

$$c_E((a_D^1, a_D^2), (a_M^1, a_M^2)) = \sum \beta_j d(a_j^A(a_D^1, a_D^2), a_j^A(a_M^1, a_M^2)) \quad \sum \beta_j = 1$$

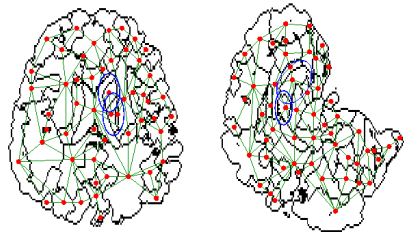
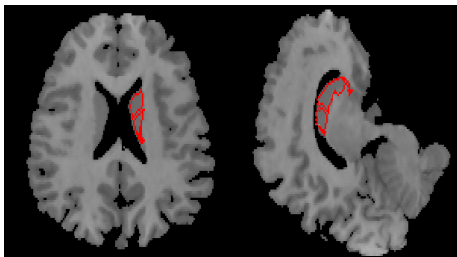
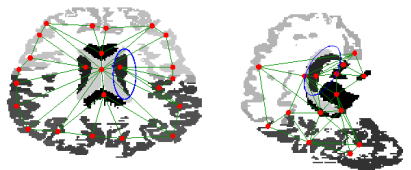
- ▶ Matching cost function h :

$$f(h) = \frac{\alpha}{|N_D|} \sum_{a_D \in N_D} c_N(a_D, h(a_D)) + \frac{1 - \alpha}{|E_D|} \sum_{(a_D^1, a_D^2) \in E_D} c_E((a_D^1, a_D^2), (h(a_D^1), h(a_D^2)))$$

Optimization methods:

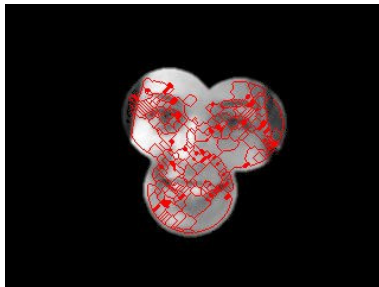
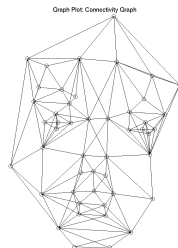
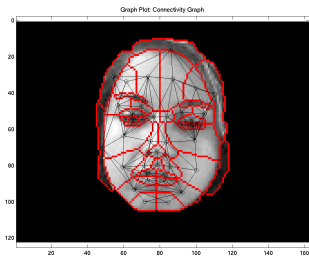
- ▶ Tree search
- ▶ Expectation Maximization
- ▶ Genetic algorithms
- ▶ ...

Example: brain structures (A. Perchant)



Example : face structures (R. Cesar et al.)

*1



Spectral method for graph matching (1)

Optimization of a cost function

- ▶ weighted adjacency matrix M
- ▶ nodes = potential assignments $a = (i, i')$ (can be selected by descriptor matching)
- ▶ edges = $M(a, b)$ agreement between the pairwise matchings a and b (geometric constraints)
- ▶ correspondance problem = finding a cluster C of assignments maximizing the inter-cluster score $S = \sum_{a, b \in C} M(a, b)$ with additional constraints
- ▶ cluster $C =$ vector x (with $x(a) = 1$ if $a \in C$ and 0 else)

$$S = \sum_{a, b \in C} M(a, b) = x^T M x$$

$$x^* = \operatorname{argmax}(x^T M x)$$

+ constraints (one to one mapping)

Spectral method for graph matching (2)

Search of the optimal cluster

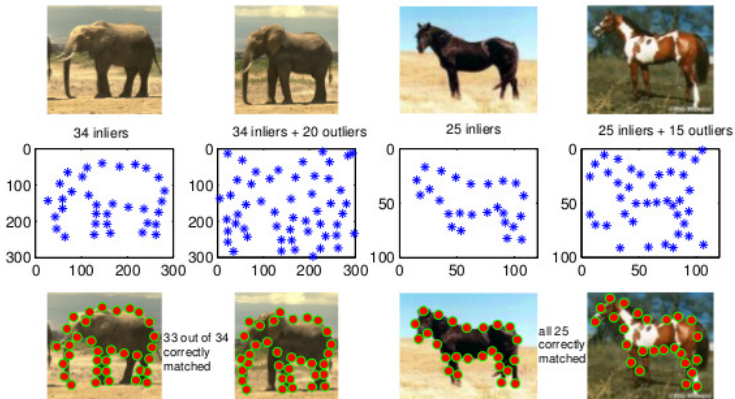
- ▶ number of assignments
- ▶ inter-connection between the assignments
- ▶ weights of the assignment

Spectral method: relaxation of the constraints on x

$$x^* = \text{principal eigenvector}(x^T M x)$$

+ introduction of the one-to-one correspondance constraints
(iterative selection of $a^* = \text{argmax}_{a \in L}(x^*(a))$
and suppression in x^* of the incompatible assignments)

Example: point matching (Leordeanu, Hebert)

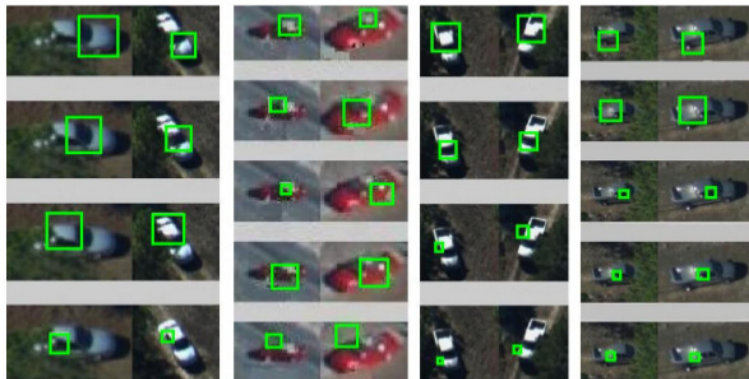


$$d_{ab} = \frac{d_{ij} + q}{d_{i'j'} + q}$$

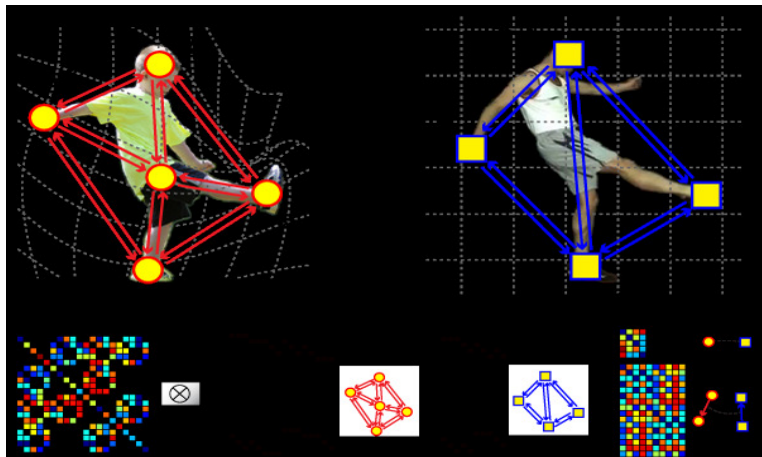
α_{ab} = angle between the matchings
(with centring and normalization)

$$M(a, b) = (1 - \gamma)c_{\alpha} + \gamma c_d$$

Example: feature matching (Leordeanu, Hebert)



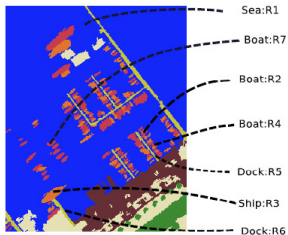
Example: factorized graph matching (Zhou, de la Torre)



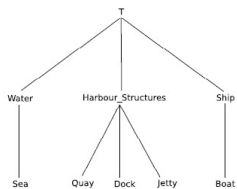
Spatial reasoning in images



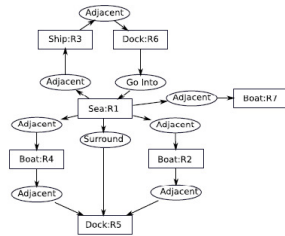
(a) Example image.



(b) Labeled image: The blue regions represent the sea, the red and orange represent ships or boats and the yellow regions represent the docks.



(c) Concept hierarchy T_C in the context of harbors.



(d) Conceptual graph representing the spatial organization of some elements of Figure 5.8(b).

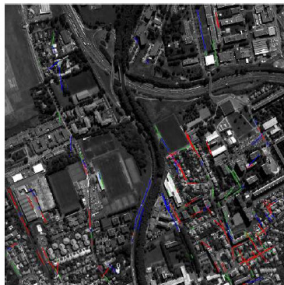
Spatial reasoning in images



(a)



(b)



(c)