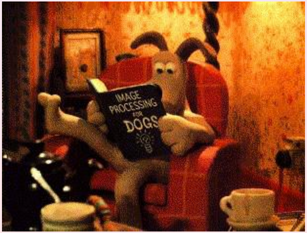
Introduction to Image Processing #2/7

Thierry Géraud

EPITA Research and Development Laboratory (LRDE)



2006

Outline

Outline

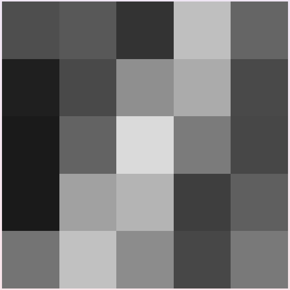
1. Image as a Graph
   - From Image to Graph
   - Sub-Graphs for the Binary Case
2. When Trees Appear
   - Connected Component Labeling
   - Distance Map
   - Trees from Image Values
3. Applications
   - Filtering
   - Segmentation
   - Cutting Graphs

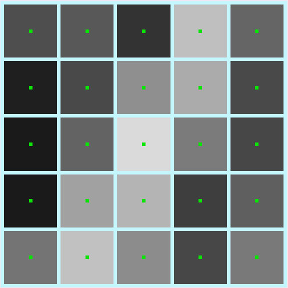
Outline

Outline

An Image (1/2)

Consider this tiny 2D gray-level image (left):





An Image (2/2)

A pixel (abrev. for “picture element”) is

- a tile in the 2D plane of an image and its associated gray value
- a couple (image, point)

The right figure:

- separates pixels with lines (cyan)
- highlights pixel centers with dots (green)

About Graph (1/3)

- An undirected graph G is an ordered pair (V, E) such as:
  - V is a set of *vertices*—or nodes,
  - E is a set of *edges*, unordered pairs of vertices.
- We say that:
  - an edge *connects* its pair of *endvertices*,
  - two vertices are *adjacent* if an edge exists between them.
- Some curiosities:
  - a *loop* is an edge whose endvertices are the same vertex,
  - an edge is *multiple* if there is another edge with the same endvertices.
- A *simple graph* is a graph with no multiple edges or loops so V and E are not multisets.

About Graph (2/3)

- The (open) *neighborhood*—or set of neighbors—of a vertex v consists of all its adjacent vertices not including v.
- A *path* is a sequence of vertices such that from each of its vertices there is an edge to the successor vertex.
- A graph is *connected* if a path can be established from any vertex to any other vertex of a graph.
- A *component* of a graph is a maximally connected subgraph.
- The *connectivity* of a graph is the minimum number of vertices needed to disconnect this graph.

About Graph (3/3)

- A *walk* is an alternating sequence of vertices and edges:
  - it is *closed* if its first and last vertices are the same,
  - it is *simple* if every vertex is incident to at most two edges,
  - it is a *cycle* if it is both closed and simple.
- The Jordan curve theorem (topology) states that every non-self-intersecting closed curve in the plane divides the plane into an ”inside” and an ”outside”.
- The *distance* between two vertices is the length of a shortest path between them.

Further Readings

- graph

  http://en.wikipedia.org/wiki/Graph_(mathematics)

- grid

  http://en.wikipedia.org/wiki/Grid_graph

- theory

  http://en.wikipedia.org/wiki/Graph_theory

- topics

  http://en.wikipedia.org/wiki/List_of_graph_theory_topics

- glossary

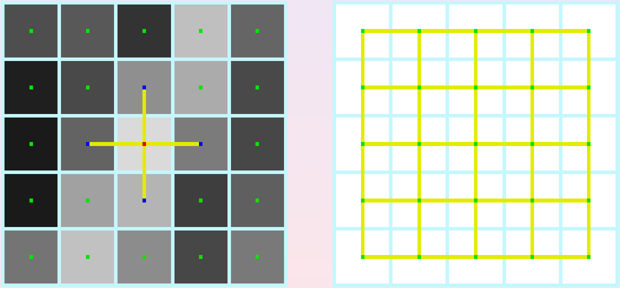
  http://en.wikipedia.org/wiki/Glossary_of_graph_theory

Turning an Image into a Graph

- Image points are natural candidate to be vertices.
- So we just need edges.
- In an image, pixels are adjacent (they touch each other) and a point has neighbors (the other points that are just around).
- Then let's go...

4-Connectivity

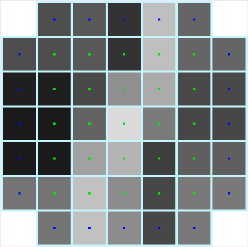
A pixel (red) has 4 neighbors (blue) and the graph is a square grid:



We have a 4-connectivity graph... except for border pixels!

Dealing with Borders (1/2)

We extend our graph outside the image domain:

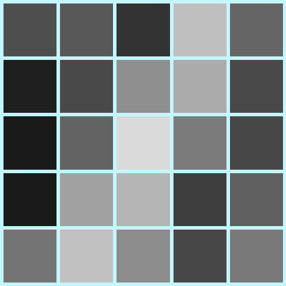


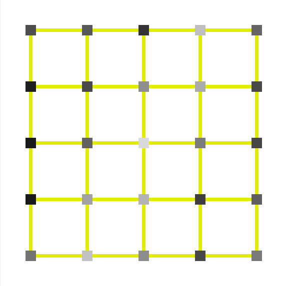
Dealing with Borders (2/2)

- Only vertices from the original domain are considered as image points (green).
- And now those vertices have 4 neighbors (blue), yet outside the image domain.

An Image as an Example

Values (gray-levels from image pixels) are associated with graph vertices.





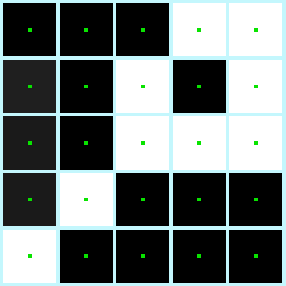
From these values, we can derive some sub-graphs and then process the image.

Outline

Binary Image

A *binary image* contains only two values:

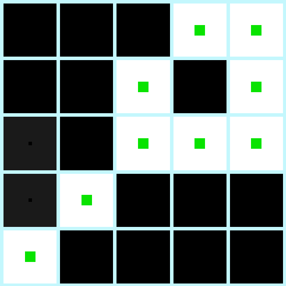
- true = white = the object
- false = black = the backgroung (not the object)

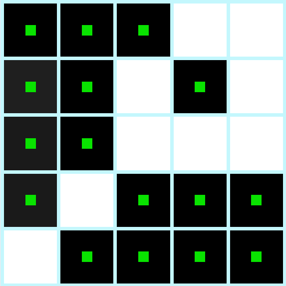


Is there one (thin) object in this image or several ones?

Sub-Graphes

We have a couple of sub-graphes, one for the object (left) and one for the background (right):

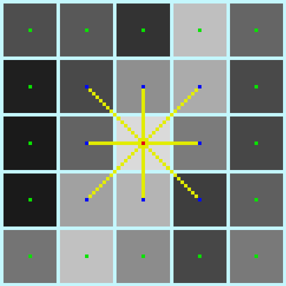


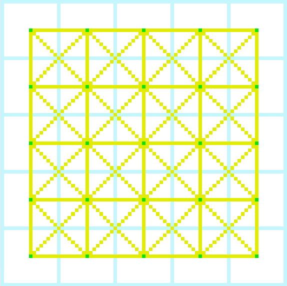


How many components do these sub-graphes have?

8-Connectivity (1/2)

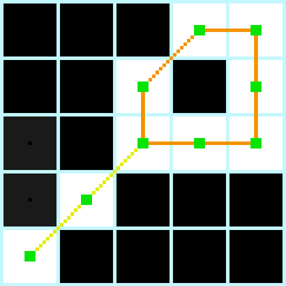
We move to 8-connectivity; now a pixel has 8 neighbors:





8-Connectivity (2/2)

Now the object sub-graph is connected—it thus has a single component:



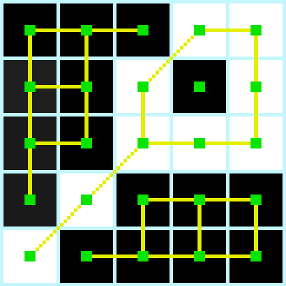
Yet, this component contains a cycle (orange) and the Jordan theorem (“inside/ouside”) do not apply!

just count the number of backgroung component given the 8-connectivity...

Solution

Always use a different connectivity for the object than for the background

either object=4 and background=8 or the contrary.



An Issue that Really Matters

That issue is of prime importance for:

- algorithms that deal with several growing components at the same time

  because a component is a well-defined region of an image

- cycles to have an inside and and ouside

  because a cycle is a contour or a propagation front in an image

Exercise 1

When the pixel is an hexagonal tile:

- what is the underlying graph?
- what is the connectivity?
- is there any problem with components and cycles?
- and what about the notion of distance?

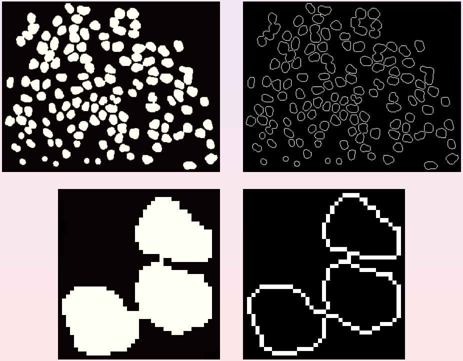
Exercise 2

If the image is 3D:

- what are the possible connectivities?
- and what connectivities are dual in the binary case?

Exercise 3

Design an algorithm that extracts the inner 8-connectivity contour of object components.



Foreword

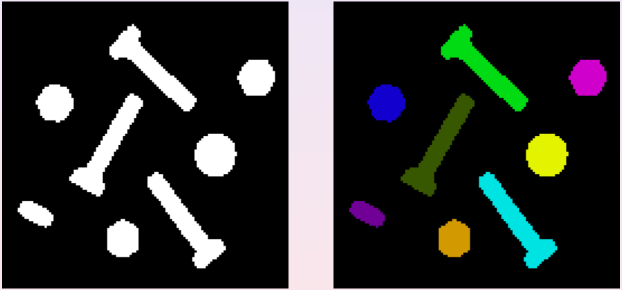
The simplest structure that defines a raw binary 2D image I is such as:

- I is a 2D matrix so, given a point $p = (r, c)$,
  - $I_{r,c}$ is a pixel value
  - $I(p)$ is another notation for this value
- at any point p, we have $I(p) \in$ *true*, *false*
- a vertex v of the object sub-graph G
  - is represented by a point of the image
  - verifies $I(v) =$ *true*
- if G is not connected, the object has several components.

Outline

State of the Problem

Consider a binary image in which the object part represents screws and bolts (left):



we want an image where every component is assigned to a label

with a particular label for the background.

State of the Problem

- each component can be represented by a spanning tree
  - this tree can be a rooted tree
  - its root node is representative
- each tree has its own label
- so we have a disjoint-set data structure—or forest of trees
- last the background is processed in a specific way

http://en.wikipedia.org/wiki/Spanning_tree_(mathematics)

http://en.wikipedia.org/wiki/Disjoint-set_data_structure

Tarjan's Union-Find Algorithm (1/2)

initialization
$l \leftarrow 0$ —CURRENT LABEL
for all p
  $L(p) \leftarrow 0$ —BACKGROUND

first pass
for all p (in video scan) such as $I(p)$ = true
  $parent(p) \leftarrow p$ —MAKE A NEW (SINGLETON) SET/TREE
  for all ante-video neighbors n of p such as $I(n)$ = true
    do_union(n, p)

second pass for all p (in reverse video scan) such as $I(p)$ = true
  if $parent(p) = p$ —ROOT POINT
    $l \leftarrow l + 1$
    $L(p) \leftarrow l$ —NEW LABEL
  else
  $L(p) \leftarrow L(parent(p))$ —LABEL PROPAGATION

Tarjan's Union-Find Algorithm (2/2)

Auxiliary functions: do_union(n, p) {

$r \leftarrow$ find_root(n)
if $r \neq p$ —TWO TREES SHOULD MERGE
$parent(r) \leftarrow p$
}

find_root(x) : point {
if $parent(x) = x$ —ROOT POINT
$\rightarrow x$
else —RECURSIVE CALL WITH TREE COMPRESSION
$\rightarrow$ find_root($parent(x)$)
}

Demo

$<$ example of a run $>$

Outline

A Need for Distances

Distance-related information is useful in practice; for instance:

- obviously the distance between two points
- the distance of a point to an object
- and also the object which a point is the closest to
- some distances between a couple of objects
- etc.

Distances (1/2)

A distance between 2 elements x and y of a set is a function to $\mathbb{R}$ which satisfies:

- $d(x,y) \geq 0$,
- $d(x,y) = 0$ if and only if $x = y$,
- $d(x,y) = d(y,x)$ (symmetry),
- $d(x,z) \leq d(x,y) + d(y,z)$ (triangle inequality).

If the set is $\mathbb{R}^n$, elements are vectors: $x = (.., x_i, ..)$ where x_i is the i^{th} coordinate of x.

The p-norm distance (Minkowsky distance of order p) is:

$$L_p(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}.$$

Distances (2/2)

We have:

Manhattan distance	$L_1(x,y) = \sum_{i=1}^{n} \lvert x_i - y_i \rvert$
Euclidean distance	$L_2(x,y) = \sqrt{\sum_{i=1}^{n} \lvert x_i - y_i \rvert^2}$
Chebyshev distance	$L_\infty = \max_{i \in [1,n]_{\mathbb{N}}} \lvert x_i - y_i \rvert$

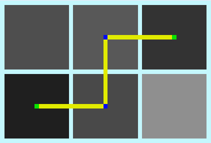
With 2D points, we can write:

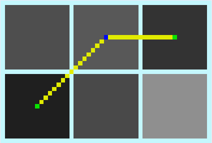
$$
\begin{aligned}
L_1(p,p') &= |r - r'| + |c - c'| \\
L_2(p,p') &= \sqrt{(r - r')^2 + (c - c')^2} \\
L_\infty(p,p') &= \max(\,|r - r'|\,,\,|c - c'|\,)
\end{aligned}
$$

http://en.wikipedia.org/wiki/Distance

Distances as Defined by Graphs (1/3)

Since we have graphs, we can directly use the notion of distance between vertices:





gives

$d = 3$ in 4-c

$d = 2$ in 8-c

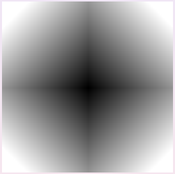
or

$d = \sqrt{2} + 1$
with Euclidean weights

whereas we have $d_{Euclidean} = \sqrt{5}$

Distances as Defined by Graphs (2/3)

With the 4-connectivity, we have the Manhattan distance (also named city-block or taxi-cab).

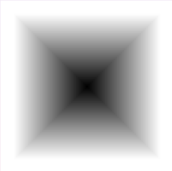


Manhattan distance to a point
(black = the point; dark and light grays = resp. short and long distances)

http://en.wikipedia.org/wiki/Taxicab_geometry

Distances as Defined by Graphs (3/3)

With the 8-connectivity, we have the chessboard distance.



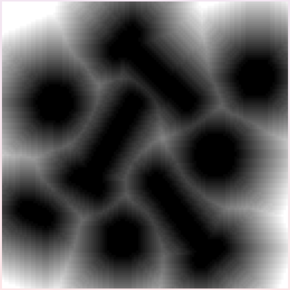
chessboard distance to a point

Both the Manhattan and the chessboard distances are quite poor approximations of the Euclidean distance.

Distance Map

A distance map is an image D whose any value $D(p)$ is the distance between p and an object.

Below the screws and bolts binary image (left) and its distance map (right):



Computing a Distance Map (1/2)

The following algorithm computes a distance map with a chamfer propagation:

initialization
for all p
$D(p) \leftarrow 0$ if $I(p) = true$, ∞ otherwise
forward pass
for all $p = (r, c)$ taken in video scan

video scan means "for each row (from up to down), for each column (from left to right), do something"

if $D(p) = \infty$
$D(p) \leftarrow min(\, D(p),\, D_{r-1,c} + 1,\, D_{r,c-1} + 1\,)$

backward pass
for all $p = (r, c)$ taken in reverse video scan
if $D(p) \neq 0$
$D(p) \leftarrow min(\, D(p),\, D_{r+1,c} + 1,\, D_{r,c+1} + 1\,)$

Computing a Distance Map (2/3)

Remark that:

- its complexity is $O(N)$ with N being the number of image pixels,
- the complexity constant factor is a multiple of the connectivity,
- this algorithm is *sequential* (contrary of parallel).

Computing a Distance Map (3/3)

Left as an exercise:

how to make this algorithm compute distances that are closer to Euclidean ones?

Computing Influence Zones (1/3)

Example:

- we have a binary image representing several objects,
- so we perform connected component labeling (left),
- and we want to obtain their influence zones (right).

Computing Influence Zones (2/3)

With L being the label image and Z the expected result:

initialization
for all p
  if $I(p) = true \quad D(p) \leftarrow 0, \;\; Z(p) \leftarrow L(p)$ else $D(p) = \infty$
forward pass
for all $p = (r, c)$ taken in video scan
  if $D(p) = \infty$
    $d \leftarrow D(p)$
    if $D_{r-1,c} + 1 < d$
      $d \leftarrow D_{r-1,c}, \;\; l \leftarrow Z_{r-1,c}$
    if $D_{r,c-1} + 1 < d$
      $d \leftarrow D_{r,c-1}, \;\; l \leftarrow Z_{r,c-1}$
    if $d \neq \infty \quad D(p) \leftarrow d, \;\; Z(p) \leftarrow l$

backward pass
left as an exercice

Computing Influence Zones (3/3)

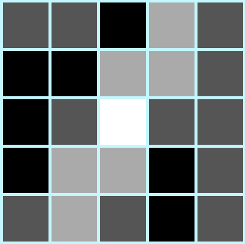
We want a map (image) so that at any point we are able to trace the shortest path towards the closest object.

- what data structure do we need for this map?
- how should we modify the previous algorithm?
- given a border point of an object, what is the underlying data structure?

Outline

Data (1/2)

Consider the following image whose values are 8-bit quantized:

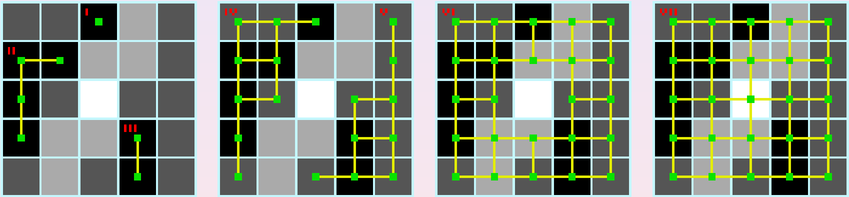


It has foor different ordered values v_i:

- black ($v_1 = 0$), dark gray ($v_2 = 85$), light gray ($v_3 = 170$), and white ($v_4 = 255$).
- for which we can derived the binary subgraphs G_i whose vertices are $\{ \, p, \, I(p) \leq v_i \, \}$.

Data (2/2)

The components of every G_i are depicted below (leftmost is G_1; rightmost is G_4):



- There is a natural inclusion: any component of G_i is included in a component of G_{i+1}.
- So all these components can be structured by an inclusion tree.

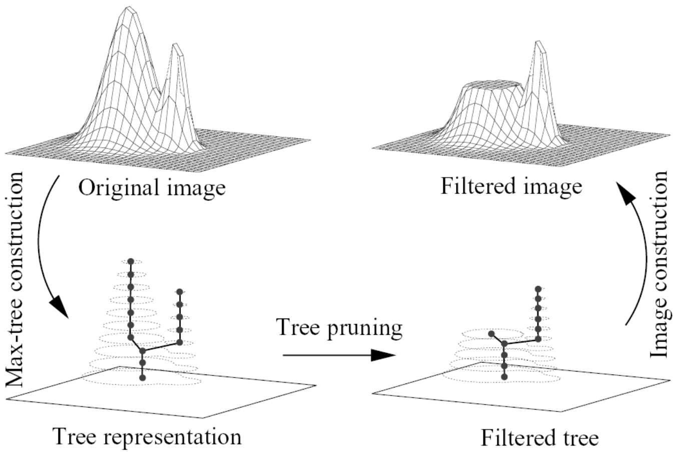
Min-Tree

This tree whose leaf nodes are the image *regional minima* is called the image *min-tree*:

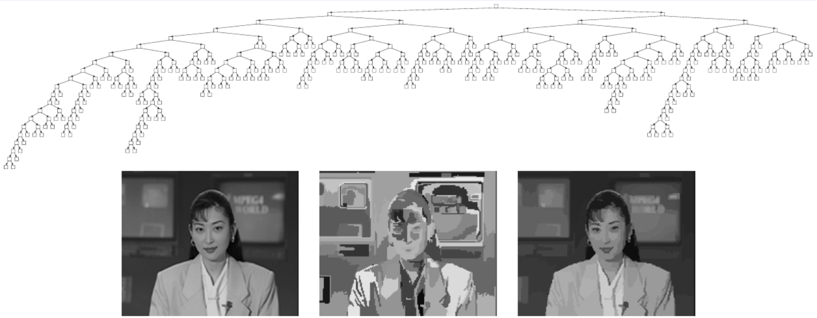
```
G4:      VII
          |
G3:       VI
        /   \
G2:    IV    V
      / \    |
G1:  I  II  III
```

A regional minima is a flat component whose outer contour is higher.

Min/Max-Tree (1/2)



Min/Max-Tree (2/2)



Further Readings (1/2)

- tree in graph theory
  `http://en.wikipedia.org/wiki/Tree_(graph_theory)`
- tree data structure
  `http://en.wikipedia.org/wiki/Tree_data_structure`
- tree traversal
  `http://en.wikipedia.org/wiki/Tree_traversal`

Further Readings (2/2)

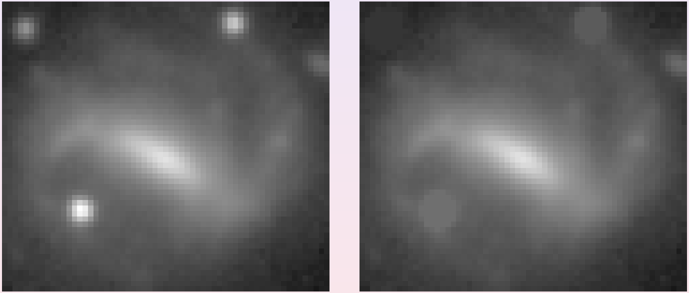
- depth-first search
  - http://en.wikipedia.org/wiki/Depth-first_search
  - http://en.wikipedia.org/wiki/Iterative_deepening_depth-first_search
  - http://en.wikipedia.org/wiki/Best-first_search
- breadth-first search

  http://en.wikipedia.org/wiki/Breadth-first_search
- A^* search algorithm

  http://en.wikipedia.org/wiki/A*_search_algorithm

Outline

Removing Stars in Galaxy Images



Identifying Boxes in Comic Strips

Remember!

when replacing the notion of pixels by the one of primitives,
e.g., regions,
we end up with high-level methods

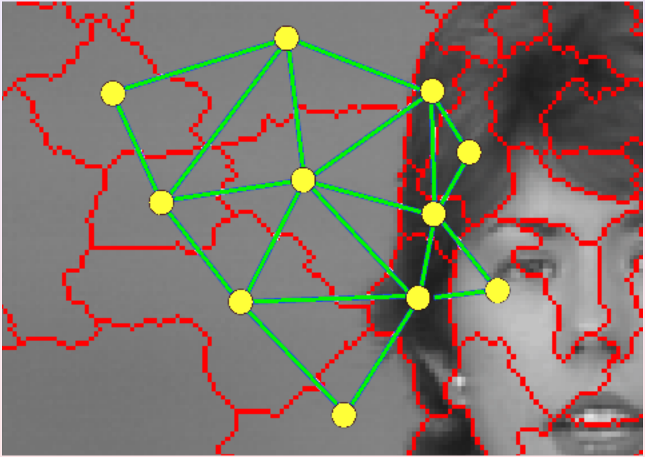
Outline

1. Image as a Graph
   - From Image to Graph
   - Sub-Graphs for the Binary Case
2. When Trees Appear
   - Connected Component Labeling
   - Distance Map
   - Trees from Image Values
3. Applications
   - Filtering
   - Segmentation
   - Cutting Graphs

Region Adjacency Graph (1/4)

Region Adjacency Graph (2/4)

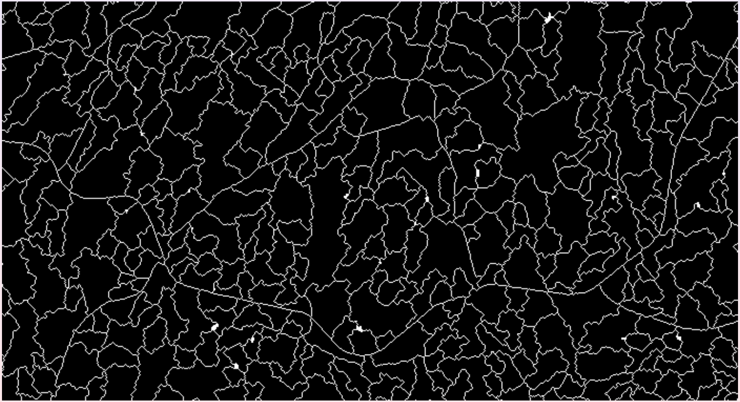
Region Adjacency Graph (3/4)



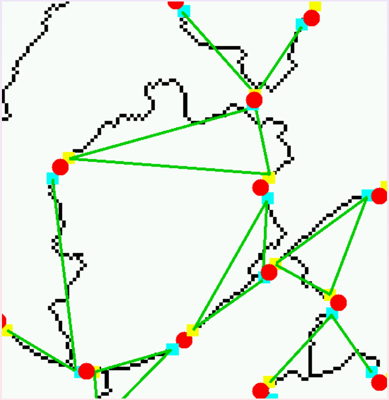
Region Adjacency Graph (4/4)

Road Identification (1/4)

Road Identification (2/4)

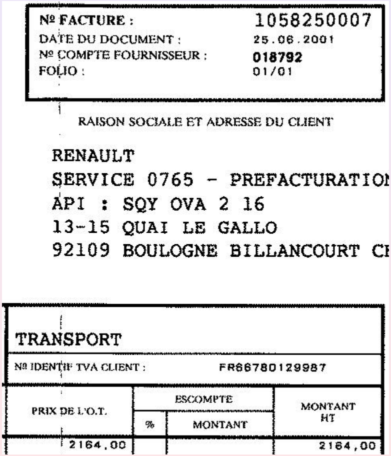


Road Identification (3/4)



Road Identification (4/4)

Text Recognition (1/8)



Nº FACTURE : 1058250007
DATE DU DOCUMENT : 25.06.2001
Nº COMPTE FOURNISSEUR : 018792
FOLIO : 01/01

RAISON SOCIALE ET ADRESSE DU CLIENT

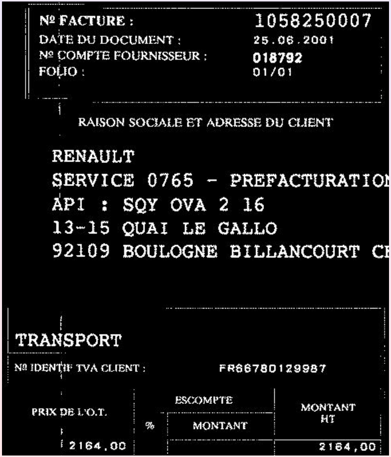
RENAULT
SERVICE 0765 - PREFACTURATION
API : SQY OVA 2 16
13-15 QUAI LE GALLO
92109 BOULOGNE BILLANCOURT CE

TRANSPORT

Nº IDENTIF TVA CLIENT : FR66780129987

PRIX DE L'O.T.	ESCOMPTE		MONTANT HT
	%	MONTANT	
2164,00			2164,00

Text Recognition (2/8)



Nº FACTURE : 1058250007
DATE DU DOCUMENT : 25.06.2001
Nº COMPTE FOURNISSEUR : 018792
FOLIO : 01/01

RAISON SOCIALE ET ADRESSE DU CLIENT

RENAULT
SERVICE 0765 - PREFACTURATIO
API : SQY OVA 2 16
13-15 QUAI LE GALLO
92109 BOULOGNE BILLANCOURT C

TRANSPORT

Nº IDENTIF TVA CLIENT : FR66780129987

PRIX DE L'O.T.	ESCOMPTE %	ESCOMPTE MONTANT	MONTANT HT
2164,00			2164,00

Text Recognition (3/8)


Nº FACTURE : 1058250007
DATE DU DOCUMENT : 25.06.2001
Nº COMPTE FOURNISSEUR : 018792
FOLIO : 01/01

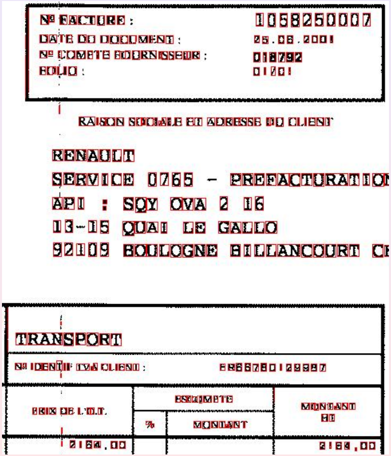
RAISON SOCIALE ET ADRESSE DU CLIENT

RENAULT
SERVICE 0765 - PREFACTURATION
API : SQY OVA 2 16
13-15 QUAI LE GALLO
92109 BOULOGNE BILLANCOURT CE

TRANSPORT
Nº IDENTIF TVA CLIENT : FR66780129987

PRIX DE L'O.T.	%	ESCOMPTE MONTANT	MONTANT HT
2164.00			2164,00

Text Recognition (4/8)


Nº FACTURE : 1058250007
DATE DU DOCUMENT : 25.06.2001
Nº COMPTE FOURNISSEUR : 018792
FOLIO : 01/01

RAISON SOCIALE ET ADRESSE DU CLIENT

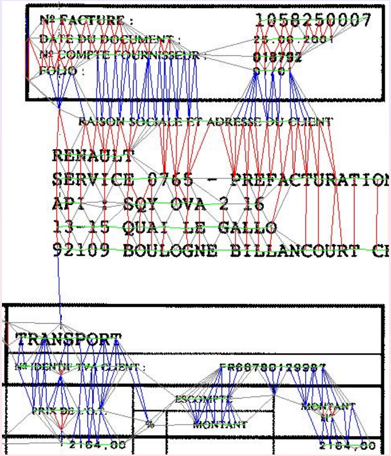
RENAULT
SERVICE 0765 - PREFACTURATION
API : SQY OVA 2 16
13-15 QUAI LE GALLO
92109 BOULOGNE BILLANCOURT CE

TRANSPORT
Nº IDENTIF TVA CLIENT : FR66780129887

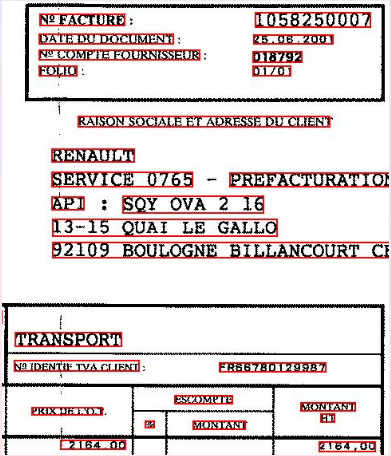
PRIX DE L'U.T.	ESCOMPTE %	ESCOMPTE MONTANT	MONTANT HT
2164,00			2164,00

Text Recognition (5/8)

Text Recognition (6/8)



Text Recognition (7/8)



Nº FACTURE : 1058250007
DATE DU DOCUMENT : 25.06.2001
Nº COMPTE FOURNISSEUR : 018792
FOLIO : 01/01

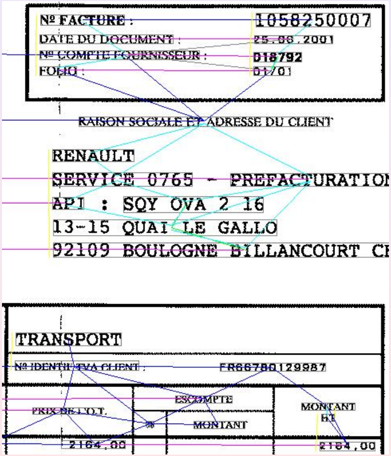
RAISON SOCIALE ET ADRESSE DU CLIENT

RENAULT
SERVICE 0765 - PREFACTURATION
API : SQY OVA 2 16
13-15 QUAI LE GALLO
92109 BOULOGNE BILLANCOURT CE

TRANSPORT
Nº IDENTIF TVA CLIENT : FR66780129987

PRIX DE LOT	ESCOMPTE %	ESCOMPTE MONTANT	MONTANT HT
2164,00			2164,00

Text Recognition (8/8)



Outline

1. Image as a Graph
   - From Image to Graph
   - Sub-Graphs for the Binary Case

2. When Trees Appear
   - Connected Component Labeling
   - Distance Map
   - Trees from Image Values

3. Applications
   - Filtering
   - Segmentation
   - Cutting Graphs

Definitions

- A *cut* is a partition of the vertices of a graph into two sets; a *cut edge* is an egde whose endvertices do not belong to the same set.
- The cut *size* is the number of its cut edges; in weighted graphs, the size is the sum of the cut edges' weights.
- A cut is a *min-cut* if the size of the cut is not larger than the size of any other cut.
- A cut is a *max-cut* if the size of the cut is not smaller than the size of any other cut.

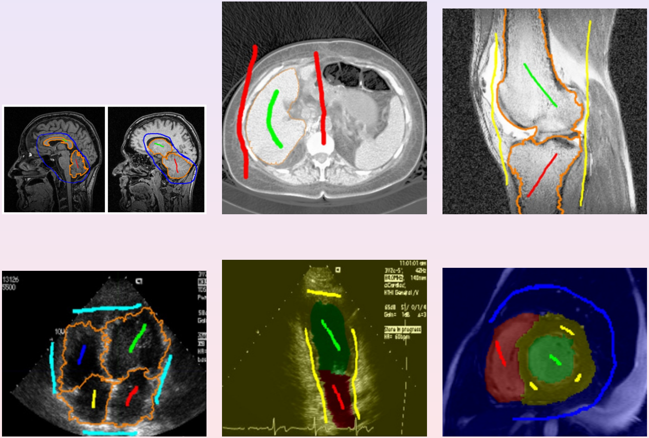
`http://en.wikipedia.org/wiki/Cut_(graph_theory)`

Properties

- Finding a min-cut can be solved by polynomial-time algorithms.
- Finding a max-cut is an NP-complete problem.
- The *m*ax-flow problem is the dual of the min-cut problem.

`http://en.wikipedia.org/wiki/Complexity_classes_P_and_NP`

Results



Leo Grady et al., Siemens Corporate Research, Princeton,

Conclusion

Having a structure (graph, tree, and so on) is often *not* enough!

We need *another* theoretical framework to put upon this structure to process images.